

ANIMA WG  
Internet-Draft  
Intended status: Experimental  
Expires: April 23, 2022

CJ. Bernardos, Ed.  
UC3M  
A. Mourad  
InterDigital  
P. Martinez-Julia  
NICT  
October 20, 2021

Autonomic setup of fog monitoring agents  
draft-bernardos-anima-fog-monitoring-05

## Abstract

The concept of fog computing has emerged driven by the Internet of Things (IoT) due to the need of handling the data generated from the end-user devices. The term fog is referred to any networked computational resource in the continuum between things and cloud. In fog computing, functions can be stitched together composing a service function chain. These functions might be hosted on resources that are inherently heterogeneous, volatile and mobile. This means that resources might appear and disappear, and the connectivity characteristics between these resources may also change dynamically. This calls for new orchestration solutions able to cope with dynamic changes to the resources in runtime or ahead of time (in anticipation through prediction) as opposed to today's solutions which are inherently reactive and static or semi-static.

A fog monitoring solution can be used to help predicting events so an action can be taken before an event actually takes place. This solution is composed of agents running on the fog nodes plus a controller hosted at another device (running in the infrastructure or in another fog node). Since fog environments are inherently volatile and extremely dynamic, it is convenient to enable the use of autonomic technologies to autonomously set-up the fog monitoring platform. This document aims at presenting this use case as well as specifying how to use GRASP as needed in this scenario.

## Status of This Memo

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

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

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

This Internet-Draft will expire on April 23, 2022.

#### Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

#### Table of Contents

1. Introduction . . . . .	2
1.1. Problem statement . . . . .	3
1.2. Fog monitoring framework . . . . .	4
1.3. Supporting simple and complex monitoring metrics . . . . .	5
2. Terminology . . . . .	6
3. Autonomic setup of fog monitoring framework . . . . .	6
4. IANA Considerations . . . . .	10
5. Security Considerations . . . . .	10
6. Acknowledgments . . . . .	10
7. Informative References . . . . .	10
Authors' Addresses . . . . .	10

#### 1. Introduction

The concept of fog computing has emerged driven by the Internet of Things (IoT) due to the need of handling the data generated from the end-user devices. The term fog is referred to any networked computational resource in the continuum between things and cloud. A fog node may therefore be an infrastructure network node such as an eNodeB or gNodeB, an edge server, a customer premises equipment (CPE), or even a user equipment (UE) terminal node such as a laptop, a smartphone, or a computing unit on-board a vehicle, robot or drone.

In fog computing, functions might be organized in service function chains (SFCs), hosted on resources that are inherently heterogeneous, volatile and mobile. This means that resources might appear and disappear, and the connectivity characteristics between these resources may also change dynamically. This calls for new orchestration solutions able to cope with dynamic changes to the resources in runtime or ahead of time (in anticipation through prediction) as opposed to today's solutions which are inherently reactive and static or semi-static.

### 1.1. Problem statement

Figure 1 shows an exemplary scenario of a (robot) network service. A robot device has its (navigation) control application running in the fog away from the robot, as a network service in the form of an SFC "F1-F2" (e.g., F1 might be in charge of identifying obstacles and F2 takes decisions on the robot navigation). Initially the function F1 is assumed to be hosted at a fog node A and F2 at fog node B. At a given point of time, fog node A becomes unavailable (e.g., due to low battery issues or the fog node A moving away from the coverage of the robot). There is therefore a need to predict the need of migrating/moving the function F1 to another node (e.g., fog node C in the figure), and this needs to be done prior to the fog/edge node becoming no longer capable/available. Such dynamic migration cannot be dealt with in today's orchestration solutions, which are rather reactive and static or semi-static (e.g., resources may fail, but this is an exceptional event, happening with low frequency, and only scaling actions are supported to react to SLA-related events).

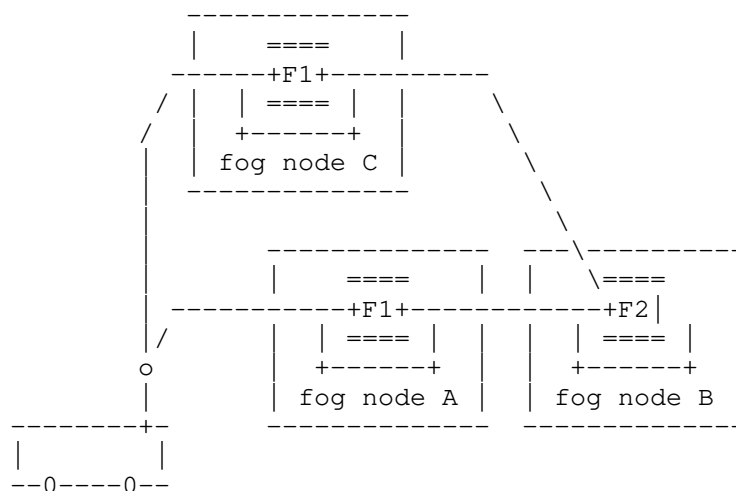


Figure 1: Example scenario

Existing frameworks rely on monitoring platforms that react to resource failure events and ensure that negotiated SLAs are met. However these are not designed to predict events likely to happen in a volatile fog environment, such as resources moving away, resources becoming unavailable due to battery issues or just changes in availability of the resources because of variations of the use of the local resources on the nodes. Besides, it is not feasible in this kind of volatile and extremely mobile environment to perform a continuous monitoring and reporting of every possible variable or parameter from all the nodes hosting resources, as this would not scale and would consume many resources and generate extra overhead.

In volatile and mobile environments, prediction (make-before-break) is needed, as pure reaction (break-before-make) is not enough. This prediction is not generic, and depends on the nature of the network service/SFC: the functions of the SFC, the connectivity between them, the service-specific requirements, etc. Monitoring has to be setup differently on the nodes, depending on the specifics of the network service. Besides, in order to act proactively and predict what might need to be done, monitoring in such a volatile and mobile environments does not only involve the nodes currently hosting the resources running the network service/service function chain (i.e., hosting a function), but also other nodes which are potential candidates to join either in addition or in substitution to current nodes for running the network service in accordance with the orchestration decisions.

In the example of Figure 1, the fog node initially hosting function F1 (fog node A) might be running out of battery and this should be detected before the node A actually becomes unavailable, so the function F1 can be effectively migrated in a time to a different fog node C, capable of meeting the requirements of F1 (compute, networking, location, expected availability, etc.). In order to be able to predict the need for such a migration and have already identified a target fog node where to move the function, it is needed to have a monitoring solution in place that instructs each node involved in the service (A and B), and also neighboring node candidate (C) to host function (F1), to monitor and report on metrics that are relevant for the specific network service "F1-F2" that is currently running.

## 1.2. Fog monitoring framework

Fog environments differ from data-center ones on three key aspects: heterogeneity, volatility and mobility. The fog monitoring framework is used to predict events triggering and orchestration event (e.g., migrating a function to a different resource).

The monitoring framework we propose for fog environments is composed of 2 logical components:

- o Fog agents running on each fog node. An agent is responsible for sending the value of a variable or parameter to a fog monitoring controller and to other fog agents. What variable or parameter will be monitored and what data will be sent (including frequency) is configured per agent considering the specifics of the network service or SFC. A fog agent might also take some autonomous actions (such as request migration of a function to a neighbor node) in certain situations where connectivity with the fog monitoring controller is temporarily unavailable.
- o A fog monitoring controller (e.g., running at the edge or at a fog node). This node obtains input from the orchestration logic (MANO stack) and autonomously decides what variables or parameters will be monitored, where will the data be collected, and how it will be done, based on the requirements provided by the orchestration logic managing the network services instantiated in the fog. This configuration is specific to a network service, a function, or an SFC as whole.
  - \* It interacts with the orchestration logic to coordinate and trigger orchestration events, such as function migration, connectivity updates, etc. In some deployments, this entity might be co-located with the orchestration logic (e.g., the NFVO).
  - \* It interacts with the fog agents to instruct what variables and/or parameters need to be monitored. It also interacts to get the resulting monitoring data. This interaction is not limited to fog agents at nodes currently involved in a given network service or SFC, but also includes other nodes that are suitable for hosting a function that needs to be migrated. This allows to provide the orchestration logic with candidate nodes in a pro-active way.
  - \* It is capable of autonomously discover and set up fog agents.

### 1.3. Supporting simple and complex monitoring metrics

Fog monitoring nodes will be capable of providing raw monitoring data as well as processed data. The former are obtained directly from the measured variables or parameters. The latter are obtained by applying some processing function to several monitoring data items. The fog monitoring controllers will specify the function to be executed, which data will be collected and processed by the functions, and the additional parameters that will control the

processing and will determine the particularities of the output of each function.

The complexity of the functions that can be executed is arbitrary. They can be either pre-instructed in the fog agents or dynamically instructed by the requester (the fog monitoring controller) by providing the sequence to execute the functions and their input parameters.

Complex monitoring metrics, the processed data, can also be used as part of the condition that determines the distributed and autonomic actions. Thus, the logic that defines those actions is simplified and the actuation components can be concentrated on their task without requiring extra effort to process the raw monitoring data.

Adding support for complex monitoring metrics enables the fog monitoring framework to avoid the transmission of unneeded data and thus optimize its overall operation. For example, if the controller is interested in the average of the CPU load of a fog agent for the last 5 minutes, it can just request it, providing the period to average as input parameter and specifying the source from which measuring the CPU load variable.

## 2. Terminology

The following terms are using in ths document:

- fog: Fog goes to the Extreme Edge, that is the closest possible to the user including on the user device itself.
- fog node: Any device that is capable of participating in the Fog. A Fog node might be volatile, mobile and constrained (in terms of computing resources). Fog nodes may be heterogeneous and may belong to different owners.
- orchestrator: In this document we use orchestrator and NFVO terms interchangeably.

## 3. Autonomic setup of fog monitoring framework

Fog nodes autonomously start fog agents at the bootstrapping, then start looking for other agents and the fog monitoring controller. This autonomic setup can be performed using GRASP. The procedure is represented in Figure 2. The different steps are described next:

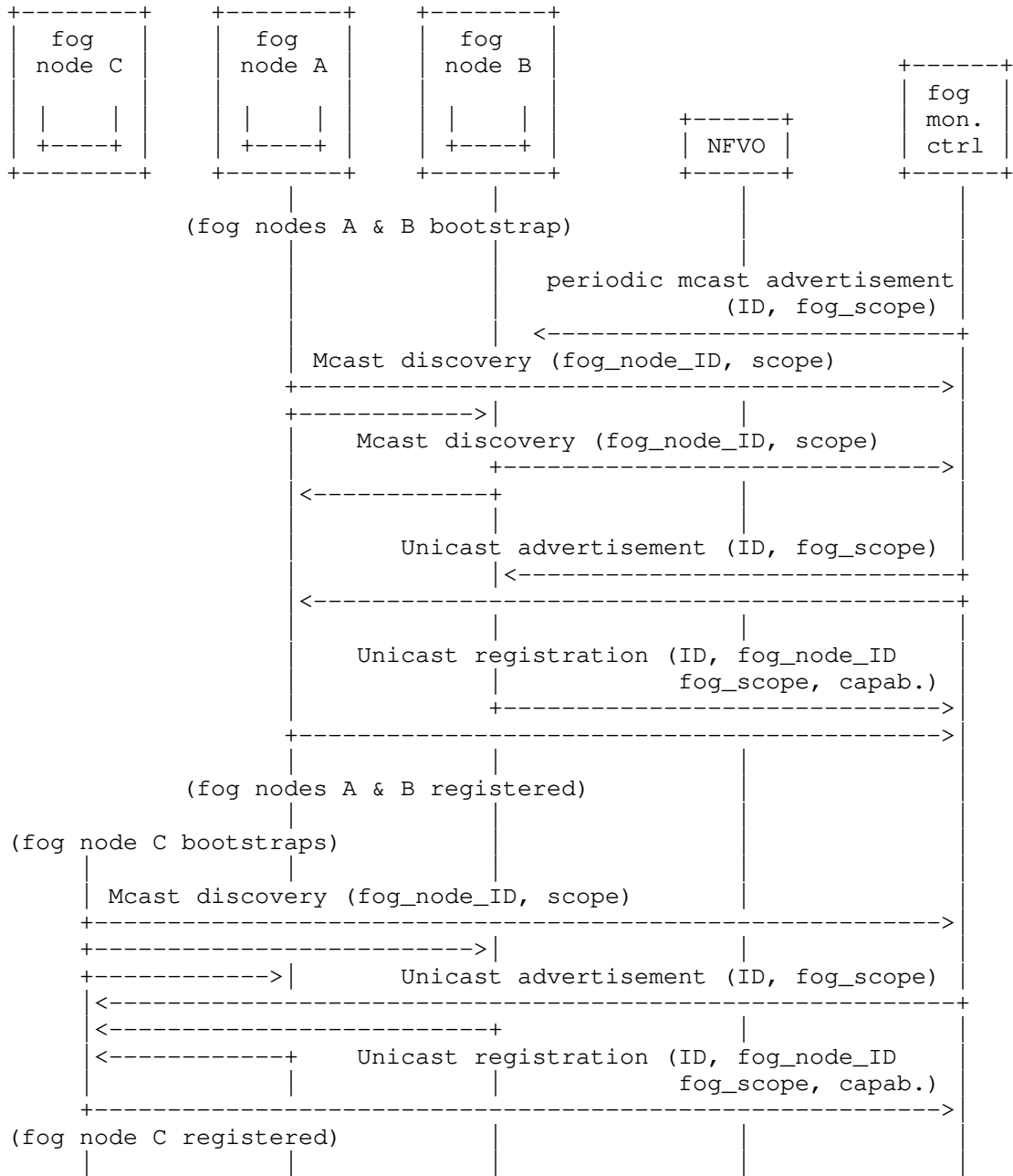


Figure 2: Autonomic setup of fog agents

- o The fog monitoring controller is regularly sending periodic multicast advertisement messages, which include its ID as well as the scope for the advertisement messages (i.e., the scope of where the messages have to be flooded).

M\_DISCOVERY messages are used, with new objectives and objective options. GRASP specifies that "an objective option is used to identify objectives for the purposes of discovery, negotiation or synchronization". New objective options are defined for the purposes of discovering potential fog agents with certain characteristics. Non-limiting examples of these options are listed below (note that the names are just examples, and the ones used have to be registered by the IANA):

- \* FOGNODERADIO: used to specify a given type of radio technology, e.g.,: WiFi (version), D2D, LTE, 5G, Bluetooth (version), etc.
- \* FOGNODECONNECTIVITY: used to specify a given type of connectivity, e.g., layer-2, IPv4, IPv6.
- \* FOGNODEVIRTUALIZATION: used to specify a given type of virtualization supported by the node where the agent runs. Examples are: hypervisor (type), container, micro-kernel, bare-metal, etc.
- \* FOGNODEDOMAIN: used to specify the domain/owner of the node. This is useful to support operation of multiple domains/operators simultaneously on the same fog network.

An example of discovery message using GRASP would be the following (in this example, the fog monitoring controller is identified by its IPv6 address: 2001:DB8:1111:2222:3333:4444:5555:6666):

```
[M_DISCOVERY, 13948745, h'20010db8111122223333444455556666',  
["FOGDOMAIN", F_SYNCH_bits, 2, "operator1"]]
```

GRASP is used to allow the fog agents and the controller discovery in an autonomic way. The extensions defined above, together with the use of properly scoped multicast addresses (as explained below), allow to precisely define which nodes participate in the monitoring and to gather their principal characteristics.

- o When a fog node bootstraps, such as nodes A and B in the figure, they start sending multicast discovery messages within a given scope, that is, the intended area that composes the fog. The definition of the scope depends on the scenario, and examples of possible scopes are:

- \* All-resources of a given manufacturer.
- \* All-resources of a given type.
- \* All-resources of a given administrative domain.
- \* All-resources of a given user.
- \* All-resources within a topological network distance (e.g., number of hops).
- \* All-resources within a geographical location.
- \* Etc.

Combination of previous scopes are also possible.

The discovery messages are multicast within the scope, reaching all the nodes that compose the specified fog resources. This can be done for example using well defined IPv6 multicast addresses, specified for each of the different scopes. This signaling is based on GRASP. Different IPv6 multicast addresses need to be defined to reach each different scope, using scopes equal or larger than Admin-Local according to [RFC7346].

- o In response to multicast fog discovery messages, the fog monitoring controller replies with unicast messages providing its information.
- o Fog agents can then register with a controller. The registration message is unicast, and includes information on the capabilities of the fog node, such as:
  - \* Type of node.
  - \* Vendor.
  - \* Energy source: battery-powered or not.
  - \* Connectivity (number of network interfaces and information associated to them, such as radio technology type, layer-2 and layer-3 addresses, etc.).
  - \* Etc.

Note that registration to multiple fog monitoring controller instances could also be possible if a fog node wants to belong to several fog domains at the same time (but note that how the

orchestration of the same resource is done by multiple orchestrators is not covered by this invention). The defined mechanisms support this via the use of fog IDs and FOGNODEDOMAIN options.

- o A fog node C bootstraps after nodes A and B are already registered. The same discovery process is followed by fog node C, but in addition to the regular advertisement, registration procedures described before, existing neighboring fog agents (such as A and B in this example), might also respond to discovery messages sent by bootstrapping nodes to provide required information. This makes the procedure faster, more efficient and reliable. In addition to helping the fog monitoring controller in the fog agent discovery process, fog agents learn themselves about the existence and associated capabilities of other fog agents. This can be used to allow autonomous monitoring by the fog agents without the involvement of the central controller.

#### 4. IANA Considerations

TBD.

#### 5. Security Considerations

TBD.

#### 6. Acknowledgments

The work in this draft will be further developed and explored under the framework of the H2020 5G-DIVE project (Grant 859881).

#### 7. Informative References

- [RFC7346] Droms, R., "IPv6 Multicast Address Scopes", RFC 7346, DOI 10.17487/RFC7346, August 2014, <<https://www.rfc-editor.org/info/rfc7346>>.

#### Authors' Addresses

Carlos J. Bernardos (editor)  
Universidad Carlos III de Madrid  
Av. Universidad, 30  
Leganes, Madrid 28911  
Spain

Phone: +34 91624 6236  
Email: [cjbc@it.uc3m.es](mailto:cjbc@it.uc3m.es)  
URI: <http://www.it.uc3m.es/cjbc/>

Alain Mourad  
InterDigital Europe

Email: Alain.Mourad@InterDigital.com  
URI: <http://www.InterDigital.com/>

Pedro Martinez-Julia  
NICT  
4-2-1, Nukui-Kitamachi, Koganei  
Tokyo 184-8795  
Japan

Phone: +81 42 327 7293  
Email: [pedro@nict.go.jp](mailto:pedro@nict.go.jp)

ANIMA  
Internet-Draft  
Intended status: Standards Track  
Expires: 12 April 2022

J. Dang, Ed.  
S. Jiang  
Huawei  
Z. Du  
China Mobile  
Z. Zhou, Ed.  
Huawei  
9 October 2021

An Autonomic Mechanism for Resource-based Network Services Auto-  
deployment  
draft-dang-anima-network-service-auto-deployment-01

Abstract

This document specifies an autonomic mechanism for resource-based network services deployment through the Autonomic Control Plane (ACP) in an Autonomic Network. This mechanism uses the GRASP in [RFC8990] to exchange the information among the autonomic nodes so that the resource among the service path can be coordinated.

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 12 April 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights

and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Requirements Language . . . . .	3
3. Terminology & Abbreviations . . . . .	3
4. Resource-based Network Services Auto-deployment Solution . .	4
4.1. ResourceManager ASA Discovery . . . . .	4
4.2. Resource Negotiation . . . . .	4
4.3. Behavior after Negotiation . . . . .	5
5. Autonomic Resource Management Objectives . . . . .	5
6. Process of Network Service Auto-deployment . . . . .	6
6.1. Process between Service Initiator and APE . . . . .	6
6.2. Process between APE and ASBR . . . . .	7
7. Compatibility with Other Technologies . . . . .	7
8. Security Considerations . . . . .	8
9. IANA Considerations . . . . .	8
10. Acknowledgements . . . . .	8
11. Normative References . . . . .	8
Authors' Addresses . . . . .	8

## 1. Introduction

With the network development, a class of services with resource requirements (such as bandwidth, latency, and jitter) are already emerging, such as video, LR, VR and so on. From network perspective, this kind of service clearly has a source IP address and a destination IP address. Therefore, once the kind of service is delivered by a network, this network service clearly has an access node and a departure node in the network. Here, the access node is called APE, and the departure node is called DPE. Actually there may be multiple Transmit nodes between APE and DPE in a network domain, and even cross multiple network domains through ASBRs. Then, the deployment of network services needs to negotiate network resources.

As is surveyed, the resource in campus network are more uneven than in the operator's network, such as wireless connections. Ideally, as a centralized system, the controller can automatically perform resource discovery and overall allocation. Actually, all devices in the campus network cannot be conveniently, availablely communicated with one controller, such as, some sensors with complex installation environment, the network devices from different vendors. Therefore, a new distributed mechanism in network is required to negotiate the resource information for network service auto-deployment.

The original purpose of this document was to validate the design of the Autonomic Networking Infrastructure (ANI) for a realistic use case. It shows how the ANI can be applied to negotiate the resource information for network service auto-deployment.

This document defines an autonomic technical objectives for Resource-based Network Services Auto-deployment. The GeneRic Autonomic Signaling Protocol (GRASP) is specified by [RFC8990] and can make use of the technical objective to provide a solution for Resource-based Network Services Auto-deployment. An important purpose of the present document is to use it for validation of the design of GRASP and other components of the ANI as described in [RFC8993].

The goal of this document is to complete the resource-based self-adaptation among service and network nodes via GRASP. And this document is not a complete functional specification of an autonomic system of Resource-based Network Services Auto-deployment, and it does not describe all detailed aspects of the GRASP objective parameters and Autonomic Service Agent (ASA) procedures necessary to build a complete system. Instead, it describes the architectural framework utilizing the components of the ANI, outlines the different deployment options and aspects, and defines GRASP objectives for use in building the system. It also provides some basic parameter examples.

## 2. Requirements Language

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.

## 3. Terminology & Abbreviations

Service Initiator(SI): It may be an end user, a Customer Edge (CE), or a controller that initiates a path-dependent and resource-based network service.

Provider Edge (PE): Provider Edge node where the network service starts or ends.

Access PE (APE): A first provider edge where the service initiator connects to the network or where the path-dependent and resource-based network service starts.

Departure PE (DPE): A last provider edge where the path-dependent and resource-based network service ends.

Transmit node: A transmit node between APE and DPE.

AS Border Router (ASBR): AS Border Router which is an edge node of the domain in the cross-domain scenario. It may also be a PE node.

#### 4. Resource-based Network Services Auto-deployment Solution

This section describes the internal architecture of resource-based network services auto-deployment. As noted in Section 1, this is not a complete description of a solution, which will depend on the detailed design of the relevant Autonomic Service Agents (ASAs). It uses the generic discovery and negotiation protocol defined by [RFC8990] and the relevant GRASP objectives are defined in Section 5.

The procedures described below are carried out by an ASA in each device that participates in the solution. We will refer to this as the ResourceManager ASA. If a device containing a ResourceManager ASA which is used up its resource, it can request more resource according to its requirements. It should decide the type and value of the requested resource and request it via the mechanism described in Section 6.

##### 4.1. ResourceManager ASA Discovery

A ResourceManager ASA that needs additional resource should firstly discover peers that may be able to provide extra resource. The ASA should send out a GRASP Discovery message that contains a ResourceManager Objective option in order to discover peers also supporting that option. Then, it should choose one such peer, most likely the first to respond.

A device that receives a Discovery message with a ResourceManager Objective option should respond with a GRASP Response message if it contains a ResourceManager ASA. If it does not contain ResourceManager ASA, the device ignore this message. Further details of the discovery process are described in [RFC8990].

##### 4.2. Resource Negotiation

After discover step, the requesting ResourceManager ASA will act as a GRASP negotiation initiator by sending a GRASP Request message with a ResourceManager Objective option. The requesting ResourceManager ASA indicates in this option the value of the requested resource. This starts a GRASP negotiation process.

When the provider ResourceManager ASA receives a subsequent Request message, it should conduct a GRASP negotiation sequence, using Negotiate, Confirm Waiting, and Negotiation End messages as appropriate. The Negotiate messages carry a ResourceManager Objective option, which will indicate the resource type and value offered to the requesting ASA.

During the negotiation, the requesting ResourceManager ASA will decide at each step how large resource need to offer. That decision, and the decision to end the negotiation, are implementation choices. As to the provider ResourceManager ASA responses how large resource they can offer and reserve enough resource during this negotiation step. A resource shortage may cause a device to indicate the existing available value within a ResourceManager Objective option to the requesting ASA. The requesting ASA compares whether the resource data received is the same locally. If they are not the same, the requesting ASA might decide whether to accept the resource. If not, the requesting ASA might terminate the negotiation via Negotiation End messages with an error code string.

As described in [RFC8990], negotiation will continue until either end stops it with a Negotiation End message. If the negotiation succeeds, the ASA that provides the resource will remove the negotiated resource from its pool, and the requesting ASA will add it. If the negotiation fails, the party sending the Negotiation End message may include an error code string.

#### 4.3. Behavior after Negotiation

Upon receiving a GRASP Negotiation End message that indicates that the acceptable resource is available. The resource providing device remove the acceptable resource from its resource pool and the requesting device may use the negotiated resource without further messages.

### 5. Autonomic Resource Management Objectives

This section defines the GRASP technical objective options that are used to support autonomic resource management.

The ResourceManager Objective option is a GRASP Objective option conforming to the GRASP specification [RFC8990]. Its name is "ResourceManager", and it carries the following data items as its value: the resource value. Since GRASP is based on CBOR (Concise Binary Object Representation) [RFC8949], the format of the ResourceManager Objective option is described in the Concise Data Definition Language (CDDL) [RFC8610] as follows:

```
objective = ["ResourceManager", objective-flags, loop-count,  
[restype, resval]]
```

```
loop-count = 0..255 ; as in the GRASP specification
```

```
objective-flags /= ; as in the GRASP
```

```
resourcetype /= 0...4; requested or offered resource type, such as  
bandwidth, latency or jitter.
```

```
resval /= 1...1000000; If the restype is bandwidth, the value ranges  
in Mbit/s; If the restype is latency, the value ranges in  
microsecond; If the restype is jitter, the value ranges in  
microsecond.
```

## 6. Process of Network Service Auto-deployment

The network service auto-deployment system includes Service Initiator, APE, DPE, and even ASBR.

The network service clearly has a APE and a DPE in the network. Actually there may be multiple Transmit nodes between APE and DPE in a single network domain, or even cross multiple network domains through ASBRs. In a single network domain, APE holds all resource information to DPE. In multiple domain network domains, APE holds all resource information to ASBR, and ASBR holds all resource information to DPE.

The Service Initiator initiates resource negotiation for a certain network service to APE. If in one single domain, APE should respond to the message with the resource valued offered. If in multiple domain network domains, APE should initiates resource negotiation to ASBR, and respond to the message with the resource valued offered until receiving ASBR's response.

### 6.1. Process between Service Initiator and APE

The Service Initiator containing a ResourceManager ASA should send out a GRASP Discovery message that contains a ResourceManager Objective option in order to discover APE also supporting that option. The APE that receives a Discovery message with a ResourceManager Objective option should respond with a GRASP Response message if it contains a ResourceManager ASA.

The ASA in the Service Initiator will act as a GRASP negotiation initiator by sending a GRASP Request message with a ResourceManager Objective option. The ASA indicates in this option the value of the requested resource.

When this ASA in the APE receives a subsequent Request message, it should conduct a GRASP negotiation sequence, using Negotiate, Confirm Waiting, and Negotiation End messages as appropriate. The Negotiate messages carry a ResourceManager Objective option with the resource value offered to the requesting ASA.

If in a single network domain, this ASA in the APE check whether the local resource data meets the requirements of the request. If it meets the requested requirement, the APE should respond with a GRASP Negotiate messages with the resource type and the resource value requested. If it doesn't meet the requested requirement, the APE should respond with a GRASP Negotiate messages with the resource value offered.

If in the multiple network domain, this ASA in the APE should act a GRASP negotiation initiator described in Section 6.2.

When the ASA in the Service Initiator receives a Negotiate message, it should check whether the resource value within the Negotiate message is the same as the resource value requested. If it is same, the Service Initiator should send GRASP Negotiation End messages indicating that the negotiation was successful. If it is not same, the Service Initiator should decide whether to accept this negotiation. If accepting this negotiation, it send should send GRASP Negotiation End messages indicating that the negotiation was successful. If not accepting this negotiation, it should send GRASP Negotiation End messages indicating that the negotiation fails.

#### 6.2. Process between APE and ASBR

The ASA in the APE should send a Confirm Waiting message to the Service Initiator, to extend its timeout. When the new resource becomes available confirmed by ASBR, the APE responds with a GRASP Negotiate message with a resource value offered.

Other processes between APE and ASBR are the same as between Service Initiator and APE.

#### 7. Compatibility with Other Technologies

A gateway device gateway device is adopted between the GRASP network and the MPLS network. As is known, the RSVP belong to the distributed mechanism for resource reservation, but it is only coupled with MPLS. Then this device uses the GRASP protocol in the GRASP network, and the MPLS protocol in the MPLS network, so that resource information can be shared.

## 8. Security Considerations

It complies with GRASP security considerations. Relevant security issues are discussed in [RFC8990]. The preferred security model is that devices are trusted following the secure bootstrap procedure [RFC8995] and that a secure Autonomic Control Plane (ACP) [RFC8994] is in place.

## 9. IANA Considerations

This document defines a new GRASP Objective option names: "ResourceManager" which is need to be added to the "GRASP Objective Names" registry.

## 10. Acknowledgements

Valuable comments were received from Michael Richardson and Brian Carpenter.

## 11. Normative References

- [I-D.ietf-mpls]  
"Multiprotocol Label Switching Architecture",  
<<https://www.rfc-editor.org/info/rfc3031>>.
- [I-D.ietf-spring-segment-routing]  
"Segment Routing Architecture",  
<<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8990] "GeneRic Autonomic Signaling Protocol (GRASP)", <<https://www.rfc-editor.org/info/rfc8990>>.
- [RFC8993] "A Reference Model for Autonomic Networking", <<https://www.rfc-editor.org/info/rfc8993>>.

## Authors' Addresses

Joanna Dang (editor)  
Huawei  
No.156 Beiqing Road  
Beijing  
P.R. China, 100095  
China

Email: dangjuanna@huawei.com

Sheng Jiang  
Huawei  
No.156 Beiqing Road  
Beijing  
P.R. China, 100095  
China

Email: jiangsheng@huawei.com

Zongpeng Du  
China Mobile  
32 Xuanwumen West St  
Beijing  
P.R. China, 100053  
China

Email: duzongpeng@chinamobile.com

Yujing (editor)  
Huawei  
No.156 Beiqing Road  
Beijing  
P.R. China, 100095  
China

Email: zhouyujing3@huawei.com

ANIMA  
Internet-Draft  
Intended status: Standards Track  
Expires: January 13, 2022

T. Eckert  
Futurewei  
M. Boucadair  
C. Jacquenet  
Orange  
M. Behringer  
July 12, 2021

Autoconfiguration of infrastructure services in ACP networks via DNS-SD  
over GRASP  
draft-eckert-anima-services-dns-autoconfig-00

Abstract

This document defines standards that enable autoconfiguration of fundamental centralized or decentralized network infrastructure services on ACP network nodes via GRASP. These are primarily the services required for initial bootstrapping of a network but will persist through the lifecycles of the network. These services include secure remote access to zero-touch bootstrapped ANI devices via SSH/Netconf with Radius/Diameter authentication and authorization and provides lifecycle autoconfiguration for other crucial services such as syslog, NTP (clock synchronization) and DNS for operational purposes.

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 January 13, 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
1.1. Overview . . . . .	2
1.2. ACP nodes supporting services autoconfiguration . . . . .	3
1.3. Use of ACP GRASP for autoconfiguration . . . . .	4
1.4. GRASP parameters . . . . .	5
2. Services . . . . .	8
2.1. Syslog . . . . .	8
2.2. NTP . . . . .	11
2.3. DNS for operations . . . . .	12
2.4. Radius . . . . .	13
2.5. Diameter . . . . .	14
2.6. SSH server . . . . .	14
3. Security Considerations . . . . .	15
4. Acknowledgments . . . . .	15
5. Change log [RFC Editor: Please remove] . . . . .	16
6. References . . . . .	16
6.1. Normative References . . . . .	16
6.2. Informative References . . . . .	17
Authors' Addresses . . . . .	17

## 1. Introduction

### 1.1. Overview

This document defines to support the autoconfiguration of Autonomic Control Plane (ACP, [RFC8994]) nodes for fundamental decentralized network services via DNS-SD GRASP, utilizing a new proposal mapping of DNS-SD ([RFC6763]) onto GRASP as its hop-by-hop multicast transport and encoding of messages.

One key purpose of this autoconfiguration is the seamless step from zero-touch bootstrap in ANI devices over to a securely remotely manageable ACP node.

Bootstrapping ANI devices via BRSKI into a running ACP can be seen as so-called "Day-0" bootstrap. If devices do not have BRSKI, then this

"Day-0" may include pre-staging of devices with the required ACP domain credentials. The mechanisms described in this document then start with what maybe could be called "Day-1" bootstrap: Auto-configuring the required functions for remote, secure access to ANI/ACP devices.

The services identified to be required for "Day-1" start with bootstrapping NTP clock synchronization across ACP/ANI nodes sufficient to validate certificates used across the ACP, establishment of user/role based authentication via Radius or diameter, autoconfiguration of the required remote-access methods to remotely access ACP/ANI nodes, SSH and NetConf with user/role based authentication. Last, but not least, in the absence of better registration mechanisms, syslog, which is also proposed to be autoconfigured via the mechanisms of this document can also serve as a "Day-1" mechanism to inform other systems about the status of ACP/ANI devices.

All autoconfiguration provided for Day-1 stays valuable and continues to operate through the lifetime of the ANI/ACP devices, so called "Day-N" services. This allows especially to change decentralized servers such as diameter/radius/NTP/syslog servers in case of failures, load-balancing or when moving devices to different locations in the network where better local server instances should be used.

[RFC8368] was written on the simple assumption that all server instances for services described in this document, NTP, Radius/Diameter, Syslog and so on are located in a so called 'Network Operations Center'. This was at the writing of [RFC8368] how this was done and called in various, mostly Enterprise networks, but is today not necessarily a good way to capture all possible deployment options. For example, server instances can today with distributed Point of Presence (POP) and edge data-centers much easier decentralized for resilience, performance and cost. Therefore, this document avoids limiting its applicability to just one "NOC" deployment option.

## 1.2. ACP nodes supporting services autoconfiguration

This document introduces the term ACPna nodes to indicate nodes supporting ACP that also support the requirements described in this document: ACP (n)oc (a)utoconfigurable.

If an ACPna node supports zero-touch bootstrap of the ACP where no configuration is possible before the ACP is enabled, then the services autoconfiguration features described in this document SHOULD be enabled by default on such an ACPna node after this zero-touch

bootstrap, because the autoconfiguration of these services can be the only method for the ACPna node to become operationally accessible from OAM systems so that it can further be configured. ANI nodes are nodes supporting ACP and BRSKI ([RFC8995]). BRSKI bootstrap is an instance of such a zero-touch bootstrap requiring auto-enablement of autoconfiguration after zero-toch bootstrap.

If an ACPna node was not zero-touch bootstrapped, then autoconfiguration SHOULD be enabled whenever ACP is enabled but may be separately configurable.

### 1.3. Use of ACP GRASP for autoconfiguration

Autoconfiguration of ACNna services utilizes the ACP instance of GRASP, ([RFC8990] as defined in [RFC8994]). It leverages and extends the GRASP objective definitions of [I-D.eckert-anima-grasp-dnssd]. Those objective elements allow to create DNS-SD compatible service announcements with flexible priority/weight and distance based selection across multiple service instances and per-service parameters.

Nodes supporting a particular service announce it via the appropriate GRASP objective into ACP GRASP. The nodes therefore need to have access to the ACP, either directly because they are ACP nodes or because they use the ACP connect function (see [RFC8994]). ACPna nodes receive these announcements and auto-configure the services tied to them. In most instances, the service announcement is from a server that a client instance on the ACPna node connects to, for example a syslog server in the POP/NOC or other location with compute. In another instance, the service is that of an authentication service and the ACPna nodes will enable a server instance that leverages the authentication service elsewhere in the network.

Note: Currently, this document does not define the option of an mDNS/DNS-SD -> ACP GRASP gateway function to enable service nodes without GRASP implementations to utilize mDNS/DNS-SD to announce their services and then expect an appropriate translation function to convert these announcements into GRASP objectives. This document does define all the GRASP objectives so that that it would be possible to define such a gateway function, but some loss of functionality would exist. For once, GRASP does support network distance based service selection (e.g., select a server from the closest service node location), whereas no such mechanism exists in DNS-SD. In addition, this document believes that support of GRASP software to announce services from service systems is very easy to accomplish.

#### 1.4. GRASP parameters

Unless otherwise described, all GRASP objective announcements described in this document SHOULD default to the following GRASP parameters. These parameters MAY all be configurable on the service nodes.

- o M\_FLOOD GRASP message, periodically sent once every 60 second. Random phase vs. full minutes (so different service announcements are distributed over time in the network).
- o ttl of 210000 msec (3.5 times 60 seconds).
- o locator-option is the ACP address of the announcing node unless the announcement is done from a third-party, for example if the announcing server does not support GRASP but GRASP is run on another service node.
- o objective-name is 'SRV.<name>', where <name> is an [RFC6335] registered service name for the service in question.
- o objective-flags is sync-only, loop-count is 255.
- o objective-value MUST comply with the requirements of [I-D.eckert-anima-grasp-dnssd].

```
[M_FLOOD, 12340815, h'fd89b714f3db0000200000064000001', 210000,
  ["SRV.syslog", 4, 255,
    { rfcXXXX: {
      &(sender-loop-count:1) => 255,
      &(srv-element:2) => {
        &(msg-type:1) => &(describe: 0),
        &(service:2)  => "syslog",
        &(instance:3) => "east-coast-primary",
        &(priority:5) => 0,
        &(weight:6)   => 65535,
        &(kvpairs:7)  => { "replicate" => 2 },
        &(range:8)   => 2,
      }
    }
  ],
  [O_IPv6_LOCATOR,
    h'fd89b714f3db0000200000064000001', TLS12, 514]
]
```

Figure 1: SRV.syslog example

The above example shows the default values for a "syslog" service announcement using the objective-value elements defined in [I-D.eckert-anima-grasp-dnssd]. SRV.syslog is the standard objective name for the "syslog" service, as is SRV.<any> for the <any> service. The announcer of this objective also provides the syslog service as it is announcing its own address in the locator option. It provides syslog on the standard syslog TCP port 514 using TLS12.

The DNS-SD equivalent service attributes are carried in the srv-element. The msg-type indicates that this objective is a service announcement. The instance of "" indicates that this service announcement for the ACP itself, and not for e.g. the data-plane. It is shown here just for illustration purposes and can be left out in encoding because it is the default. Likewise, the service element is redundant and shown only for illustrative purposes. Priority and weight have the same semantic as in DNS-SD SRV records. In this case, the service announcement indicates the highest priority (0) and the highest weight (65535). Kvpairs includes service specific options.

Going beyond the capabilities, the range parameter indicates that the client of this service should select this announced service not only by priority/weight but primarily by the distance in terms of network hop-count between this service announcer and the client: The client is expected to select the best service announcement by priority and weight only between alternatives that are not more than two network hops apart in distance to the client. Otherwise the client should pick the closer one.

To allow the client to know the distance to a service announcement, the sender-loop-count parameter is included in the announcement. It MUST be set by the sender to the same value (255 in this example) as the loop-count in the GRASP header. The loop-count in the header is hop-by-hop reduced. When the GRASP message arrives at the client, the difference between sender-loop-count and loop-count is the distance to the service announcer in hops.

```

;
; Following GRASP header definitions from GRASP
;
flood-message = [M_FLOOD, session-id, initiator, ttl,
                 +[objective, (locator-option / [])]]
objective = ["SRV.<rfc6335-name>", objective-flags, loop-count,
            objective-value]

objective-flags = sync-only ; as in GRASP spec
sync-only      = 4          ; M_FLOOD only requires synchronization
loop-count     = 255        ; recommended
;
; Following GRASP objective-value definitions from GRASP DNS-SD
;
objective-value = { 1*elements }
elements        = ( @rfcXXXX: { 1*relement } )
relement       //= ( &(sender-loop-count:1) => 1..255 )
relement       //= ( &(srv-element:2) => context-element )
context-element = {
    ?( &(private:0)      => any),
    ?( &(msg-type:1)     => msg-type),
    ?( &(service:2)      => tstr),
    *( &(instance:3)     => tstr),
    ?( &(domain:4)       => tstr),
    ?( &(priority:5)     => 0..65535 ),
    ?( &(weight:6)       => 0..65535 ),
    *( &(kvpairs:7)      => { *(tstr: any) },
    ?( &(range:8)        => 0..255 ),
    *( &(clocator:9)     => clocator),
}
;
TLS12 = 257

```

Figure 2: GRASP service definition

The above picture shows the complete CDDL definition of a GRASP M\_FLOOD message indicating a service together with the objective-value encoding. Som of the context-element options are not used in this document (TBD - remove before going RFC).

The value 257 is defined to indicate TLS12 ([RFC5246]) to be used in the protocol field of GRASP locators to indicate that a TCP port is intended to be used with TLS version 1.2. Values 1..255 are reserved for IP protocol numbers.

## 2. Services

### 2.1. Syslog

ACPna nodes SHOULD support autoconfiguring of syslog via the SRV.syslog objective.

When an ACPna node discovers one or more SRV.syslog announcements across the ACP, it SHOULD perform syslog operations to the best available discovered server.

Local configuration of syslog on the ACPna node SHOULD have no impact on the autoconfigured syslog operations, or else, misconfiguration could cause to failure of the autoconfigured syslog operations. Instead, configured syslog operations should just operate as ships-in-the-night to the GRASP learned autoconfigured syslog operations.

Severity of syslog messages SHOULD be 5 (Notice) (see [RFC5424]), and all messages that are necessary to support normal remote operations of the node should be assigned severities higher (numerically lower) or equal to 5/Notice.

Syslog service announcements SHOULD include the instance option, indicating the unique name of the service instance described by the GRASP objective. This serves diagnostics and avoids having to identify service instances by the address(es) in the locator-options. In the example Figure 1, the instance name is "east-coast-primary".

The syslog facility value is a choice of the ACPna node, the autoconfigured syslog server must be able to deal with any syslog facility code received. If an ACPna node has no pre-established standard for the facility-code, then the value of local7 (23) MAY be used.

For resilience, it may be appropriate to receive syslog messages on more than one server. A server can indicate this via the "replicate" keyword in the GRASP objective-value kvpair element. The value of the "replicate" keyword indicates the maximum number of syslog servers that the client SHOULD autoconfigure syslog to. After selecting the best service announcement, the client looks up the value N of the "replicate" keyword of that best servers announcement and selects the best N-1 service announcements and ultimately logs to all N. ACPna nodes SHOULD support autoconfigured syslog to up to 3 servers simultaneously.

Autoconfigured syslog SHOULD support TLS1.2, TCP and UDP. Because ACP provides encryption, use of just TCP instead of TLS should be sufficient and may achieve higher performance. Use of UDP should be

avoided because of the potential to loose packets and not supporting congestion control.

If a syslog server supports more than one transport option (TLS1.2, TCP, UDP), it SHOULD announce them via a single GRASP objective and list them via clocator options of the srv-element because the locator-option in the GRASP header (as shown in example Figure 1) allows only one locator-option. The order of the clocator options in the indicates the preference of the server. From this list, the client supports the first option supported also by the client and ignore the others. The context of the clocator would normally be "", indicating that the locator-option address is reachable via the ACP.

Instead of (or in addition to) using multiple clocator options, a server can also announce multiple SRV.syslog objectives, but in that case each of them would be considered to be a different service instance considered by the the client when selecting the (set of) best service instances. If a service announcement indicates via the "replicate" keyword that the client should log to three service instances, and announce three separate SRV.syslog objectives, each one with a different locator-option, then the client might select to log to all three of them - instead of - which is more likely the desired option - for the client to log to actually three different servers. Hence the use of multiple clocator options that are examined by clients only after server selection is done.

When a client uses TLS, it MUST use its ACP domain certificate for authentication. Likewise, the syslog server MUSTS use its ACP domain certificate.

Logging by default uses the ACP, in the clocator option, this is indicated via a context value of "". Servers may also indicate support for logging across the data-plane, which may provide higher performance but may fail if reachability in the data-plane does not exist, so care must be taken when announcing this option. For example, in managed MPLS/VPN networks where the ACP extends across P/PE and CE devices, the global routing table on a CE device is often not the same as that on P/PE devices, and therefore CE devices may not be able to log to "0". In this case, the syslog server should instead announce a deployment choosen name for the context, such as "VRF0". Clients would only take such a clocator into account if there is a local configuration that maps the context name to a routing table. In this example, only P/PE nodes would have this configuration, therefore allowing the CE nodes to ignore this clocator; And if this clocator was the only locator-option in the GRASP objective, then the whole objective MUST be ignored by the client when selecting the best possible service instance. Note that for contexts other than the ACP (""), both IPv4 and IPv6 are

possible, depending on what version(s) of IP are deployed in the data-plane.

Failure to connect to a chosen service instance SHOULD be taken into accounts by clients when selecting service instances to log to. For UDP locator-options, ICMP/ICMPv6 error indications are such connection failures. For TCP/TLS connections, connection failure includes TCP and TLS failures as well as keepalive failures. When failures occur, clients should attempt to re-connect with exponential timeouts, starting with 5 seconds and staying at 320 seconds or until the GRASP service announcement expires and is not refreshed.

When connecting to a server fails, the ACPna client SHOULD connect to the next best available server in the meantime. ACPna client SHOULD support connecting to up to four service instances if any connections fail. If for example the client is logging to two service instances because 2 is indicated in the "replicate" option of the service announcements, and one fails, the client will attempt to re-connect to it while in parallel establishing syslog connection to a third-best service-instance.

When establishing connection to a new syslog service instance, ACPna clients SHOULD log with severity 5 an indication of this event, indicating its own ACP address, the ACP address and if existing instance name of the new syslog service instance and the reason. Like any other autoconfigured syslog message, this would go to all syslog connections and therefore show up on the redundant syslog servers, allowing to recognize failure of connectivity to another syslog server - and tracing of client logs across syslog servers if the client changes them.

Examples:

```
ACP: fd89:b714:f3db::0200:0000:6400:0042 start logging to:
fd89:b714:f3db::0200:0000:6400:0001/east-coast-primary,TLS reason:
starting up
```

```
ACP: fd89:b714:f3db::0200:0000:6400:0042 start logging to:
fd89:b714:f3db::0200:0000:6400:0001/east-coast-primary,TLS reason:
new better service instance
```

```
ACP: fd89:b714:f3db::0200:0000:6400:0042 stop logging to:
fd89:b714:f3db::0200:0000:6400:0001/east-coast-secondary,TLS reason:
connection failure
```

When failures to deliver syslog messages to ANY syslog servers happen, clients SHOULD track the this and indicate loss of messages via the next working syslog connection. Note that due to the

possibility of ICMP/ICMPv6 errors, only the successful delivery of messages via TLS or TCP should be tracked. TBD: need to check if this can reasonably be recommended, pending on availability of e.g. TAPS API spec to know whether a TCP write was sent and acknowledged by the receiver (given how there are no reply messages in syslog).

## 2.2. NTP

Time synchronization is one of the most fundamental functionality for network devices for a variety of functions to work and also for diagnostics to be comparable across the network. If problems propagate fast across the network, the client generated timestamp of events in syslog messages (or other diagnostics function) allows to trace event propagation and deduce causality. This may require network clock synchronization in the order of milliseconds, something which is easily achievable in today's network devices via NTP.

ACPna nodes SHOULD support autoconfiguration of clock synchronization through NTP ([RFC5905]) with the following autoconfiguration semantics.

The GRASP objective for NTP is SRV.ntp. This does not distinguish between NTPv4 and NTPv3 because NTPv4 is fully backward compatible with NTPv3, so server and client will negotiate between these two versions.

The kvpair key "stratum" has a numeric value and indicates the stratum or level of a server in a synchronization tree. The value of 1 indicates the root of the distribution tree. Servers that synchronize from the master have a stratum of 2, and so on.

The kvpair key "minpoll" indicates the lowest periodic polling that the client will perform against the server. Announcing a large numeric value allows for a server to reduce the amount of NTP messages from clients, but slows down convergence time of clients number of service instances that simultaneously bootstrap.

The kvpair key "key" indicates the NTPv3 authentication mechanism. When present, clients MUST use the value as the key to perform NTPv3 (MD5) hash authentication of message with this service instance. Note that the encryption of the ACP serves as protection of distributing such a cleartext symmetric key via GRASP to clients.

TBD: Understand NTPv4 autokey and define appropriate kvpair to enable auto-configuring it, especially when the service instance announcement indicates the use of the data-plane.

The autoconfiguration described in the following paragraphs is for leafs of the clock distribution graph, e.g., nodes that do only aim to obtain synchronized time from a server. Configuration of the server hierarchy is left to explicit configuration.

Clients SHOULD select service instance(s) with the worst (highest) stratum value. In the face of multiple equal options, clients have to pick the best ones based on the standard selection criteria priority/weight and range, allowing for distributed NTP server deployment by e.e., setting range to 1, or via centralized deployment with multiple servers, setting range to 255 and priority/weight accordingly. Making the stratum the primary selection criteria allows in the future to also introduce autoconfiguration of servers in the NTP clock distribution tree without incurring the problem that a large number of clients would then select higher stratum servers (and overload them).

Like most other autoconfigured services, the autoconfigured NTP time synchronization SHOULD take precedence over explicit configured NTP options to ensure that time synchronization is not subject to misconfiguration of individual nodes (but only subject to misconfiguration of servers).

The kvpair "TZ" option allows to signal the time zone of the ACP network to clients. Its value is a string indicating the time zone of all nodes in the ACP network. Care must be taken not to use this option in networks extending across multiple time zones. Because time zone distribution does not work automatically across larger networks with multiple time zones, overriding the signalled time zone SHOULD be possible through local configuration.

TBD: references for time zone spec standards and also for DST rule indications.

### 2.3. DNS for operations

Availability of DNS names for network operations/troubleshooting is today mostly an convenience in network operations, but with IPv6 evolving the need to use DNS names even in CLI based network diagnostics is raising - because IPv6 addresses often are more difficult to memorize by operators. More and more network features also support configuration that instead of addresses include domain names or URLs, and ultimately, any non-fully autoconfigured functions should rather rely on domain-names and URLs instead of just addresses for greater flexibility and reliability in the face of address changes.

In the face of this, ACPna nodes SHOULD support autoconfiguration of DNS for operational purposes. "For operation purposes" implies that the use of the autoconfigured DNS servers SHOULD NOT be used for DNS services offered to users of the data plane, such as DNS proxy services. This would cause the ACP to effectively carry user traffic, whenever a client DNS request to an ACPna node with a DNS proxy would be forwarded to an autoconfigured server via the ACP.

The GRASP objective name for such OAM use of DNS is OAM-DNS. It is explicitly not SRV.dns to highlight that this instance of DNS is copied for operational purposes only to isolate it from user issues (performance across the ACP and attacks). Utilizing different DNS contexts also allows to set up split-horizon DNS where all the operationally relevant DNS names are only made available via the DNS servers or zones available across the ACP.

The value of the "search-list" kvpair option is a ";" (semicolon) separated list of domain name prefixes that should be searched by the client for non-FQDN that they need to resolve. "local-arpa" is the prefix to use for reverse IPv4/IPv6 address lookups. If for example "local-arpa" is set to "arpa.example.com", then the clients should first look up IPv4/IPv6 addresses in "ipv6.arpa.example.com"/"in-addr.arpa.example.com" before resorting to lookup in the Internet global "ipv6.arpa"/"in-addr.arpa". For RFC1918/ULA addresses, no fallback to the global reverse lookup prefixes should be done.

ACPna nodes SHOULD look up their name via a reverse lookup of their ACP address, and then auto-configure this name.

There are no service specifics for the selection of DNS servers. A ACPna node simply uses the standard priority/weight/range options to select a DNS server. It MAY prefer a server with TCP locator-option simply because that allows in most cases faster discovery of connectivity problems than a UDP connection.

TBD: Note that it is fairly easy to re-use the autoconfiguration scheme described here to provide auto-configuration of DNS for user DNS services with the help of the ACP. The objective name would have to be changed and the clocators would have to indicate a data-plane context, so that user requests are carried across the data-plane from DNS proxies to DNS servers. It is unclear if this service should be described in this document though.

## 2.4. Radius

Radius [RFC2865] is a protocol used for AAA service - Authentication, Authorization and Accounting. Autodiscovery of Radius servers across the ACP for ACPna nodes serves the purpose to enable authentication

and authorization of other ACPna autoconfigured services such as below described Section 2.6.

ACPna nodes MUST support Radius and/or Diameter autoconfiguration if they support any of the autoconfigured services depending on such an authentication service.

The GRASP objective name is SRV.radius. The UDP or TCP port of the locator-option in the GRASP header or the clocator option indicate the UDP or TCP port of the Radius servers authentication connection. The context of a clocator MUST be "" to indicate the ACP - because the Radius connections MUST pass across the ACP to be protected against eavesdropping - and the radius security methods described here are not sufficiently secure to allow passing them across the data-plane.

The kvpair "secret\_key" string value indicates the secret key to use on the connection to the Radius server. The optional "acct\_port" numeric value indicate the UDP/TCP port of any accounting connection supported by the radius server. The protocol (UDP vs. TCP) is the same as the one in the chosen locator-option/clocator.

There are no service specific selection rules. TCP is preferred for faster recognition of a failed server and reselection of an alternative server.

The specific data/authentication/authorization configuration required on the Radius server depends on the OAM service authenticated/authorized and is described in its section in this document.

TBD: Should we define AVpair or different objective names to distinguish what services can be authenticated? Would be easier if we found another service than SSH/Netconf.

## 2.5. Diameter

TBD. Alternative to Radius. Author would welcome suggesting what parameters are relevant for a diameter authentication service.

## 2.6. SSH server

ACPna nodes supporting SSH server functionality for remote management access via CLI, NETCONF or other methods SHOULD auto-enable SSH server functionality across the ACP whenever they are aware from ACP GRASP of RADIUS (Section 2.4) or DIAMETER (Section 2.5) authentication servers. ACPna nodes that support ACPna SSH server functionality MUST support authentication via either RADIUS and/or Diameter.

If both protocols are supported by the ACPna node, the ACPna node SHOULD select the authentication server based on the service priority parameters across both protocols. E.g., if a RADIUS server has a higher priority in GRASP than the DIAMETER server, the ACPna node should authenticate against the RADIUS server.

When valid authentication server(s) are discovered, the SSH server is autoconfigured. It SHOULD only listen to the standard SSH port with the ACP address of the node but not be reachable from the data-plane. It MUST NOT be modifyable by configuration (only by auto-configuration). If autoconfiguration of an SSH server on the standard SSH port conflicts with explicitly configured SSH server for the data-plane due to software limitations or complexity, the autoconfigured SSH server MAY be started on a node-type specific and not dynamically selected port number. This port number must be well-known to OAM operations as there is no method provided to signal it to the SSH client side.

Note that this document does not define any standards for the exact message options for authentication or authorization. Especially authorization, such as privilege level that permits to change configuration is likely using vendor specific methods, and Radius/Diameter servers must be capable to recognize the type of client as they had to without this autoconfiguration.

### 3. Security Considerations

There is no protection against "unauthorized" ACP nodes to generate service announcements, because there is no authorization scheme in GRASP. Discovery of unauthorized announcers is easy though because the service announcements are flooded across the ACP and are therefore easily visible on nodes that may specifically observe announcements to discover unauthorized ones.

A possible framework to define authorization could rely on defining roles for ACP nodes either through additional parameters in their ACP domain certificate or following initial provisioning, and then lock down the ability for later configuration to enable services (and their GRASP announcements) to only those included in the role assigned to the node. This is outside the scope of this document.

### 4. Acknowledgments

Thanks to Ignas Bagdonas for deployment / applicability / terminology input and to Balaji BL, Ravi Kumar Vadapalli for their original implementation of the concept.

## 5. Change log [RFC Editor: Please remove]

### draft-eckert-anima-services-dns-autoconfig

00: Renaming from 'noc-autoconfig' after a long discussion with Ignas Bagdonas: replaced all mention of NOC with "infrastructure / decentralized" services, because the term NOC seems to be a terminology that does not well match how it is called in many type of networks.

### draft-eckert-anima-noc-autoconfig (2018)

00: Initial version.

## 6. References

### 6.1. Normative References

- [I-D.eckert-anima-grasp-dnssd]  
Eckert, T., "DNS-SD compatible service discovery in GRASP", draft-eckert-anima-grasp-dnssd-01 (work in progress), July 2018.
- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <<https://www.rfc-editor.org/info/rfc2865>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <<https://www.rfc-editor.org/info/rfc5424>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <<https://www.rfc-editor.org/info/rfc6335>>.

- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC8368] Eckert, T., Ed. and M. Behringer, "Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM)", RFC 8368, DOI 10.17487/RFC8368, May 2018, <<https://www.rfc-editor.org/info/rfc8368>>.
- [RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <<https://www.rfc-editor.org/info/rfc8990>>.
- [RFC8994] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "An Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <<https://www.rfc-editor.org/info/rfc8994>>.

## 6.2. Informative References

- [RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <<https://www.rfc-editor.org/info/rfc8995>>.

## Authors' Addresses

Toerless Eckert  
Futurewei Technologies USA  
Inc.  
2220 Central Expressway  
Santa Clara 95050  
USA

Email: [tte+ietf@cs.fau.de](mailto:tte+ietf@cs.fau.de)

Mohamed Boucadair  
Orange  
Rennes 35000  
France

Email: [mohamed.boucadair@orange.com](mailto:mohamed.boucadair@orange.com)

Christian Jacquenet  
Orange

Email: christian.jacquenet@orange.com

Michael H. Behringer

Email: michael.h.behringer@gmail.com

ANIMA  
Internet-Draft  
Intended status: Standards Track  
Expires: 5 September 2022

T.T.E. Eckert  
Futurewei  
M. Boucadair  
C. Jacquenet  
Orange  
M. Behringer  
4 March 2022

Autoconfiguration of infrastructure services in ACP networks via DNS-SD  
over GRASP  
draft-eckert-anima-services-dns-autoconfig-01

Abstract

This document defines standards that enable autoconfiguration of fundamental centralized or decentralized network infrastructure services on ACP network nodes via GRASP. These are primarily the services required for initial bootstrapping of a network but will persist through the lifecycles of the network. These services include secure remote access to zero-touch bootstrapped ANI devices via SSH/Netconf with Radius/Diameter authentication and authorization and provides lifecycle autoconfiguration for other crucial services such as syslog, NTP (clock synchronization) and DNS for operational purposes.

Status of This Memo

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

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

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

This Internet-Draft will expire on 5 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	2
1.1. Overview . . . . .	2
1.2. ACP nodes supporting services autoconfiguration . . . . .	3
1.3. Use of ACP GRASP for autoconfiguration . . . . .	4
1.4. GRASP parameters . . . . .	5
2. Services . . . . .	8
2.1. Syslog . . . . .	8
2.2. NTP . . . . .	11
2.3. DNS for operations . . . . .	12
2.4. Radius . . . . .	14
2.5. Diameter . . . . .	14
2.6. SSH server . . . . .	15
3. Security Considerations . . . . .	15
4. Acknowledgments . . . . .	16
5. Change log [RFC Editor: Please remove] . . . . .	16
6. References . . . . .	16
6.1. Normative References . . . . .	16
6.2. Informative References . . . . .	17
Authors' Addresses . . . . .	18

## 1. Introduction

### 1.1. Overview

This document defines to support the autoconfiguration of Autonomic Control Plane (ACP, [RFC8994]) nodes for fundamental decentralized network services via DNS-SD GRASP, utilizing a new proposal mapping of DNS-SD ([RFC6763]) onto GRASP as its hop-by-hop multicast transport and encoding of messages.

One key purpose of this autoconfiguration is the seamless step from zero-touch bootstrap in ANI devices over to a securely remotely manageable ACP node.

Bootstrapping ANI devices via BRSKI into a running ACP can be seen as so-called "Day-0" bootstrap. If devices do not have BRSKI, then this "Day-0" may include pre-staging of devices with the required ACP

domain credentials. The mechanisms described in this document then start with what maybe could be called "Day-1" bootstrap: Auto-configuring the required functions for remote, secure access to ANI/ACP devices.

The services identified to be required for "Day-1" start with bootstrapping NTP clock synchronization across ACP/ANI nodes sufficient to validate certificates used across the ACP, establishment of user/role based authentication via Radius or diameter, autoconfiguration of the required remote-access methods to remotely access ACP/ANI nodes, SSH and NetConf with user/role based authentication. Last, but not least, in the absence of better registration mechanisms, syslog, which is also proposed to be autoconfigured via the mechanisms of this document can also serve as a "Day-1" mechanism to inform other systems about the status of ACP/ANI devices.

All autoconfiguration provided for Day-1 stays valuable and continues to operate through the lifetime of the ANI/ACP devices, so called "Day-N" services. This allows especially to change decentralized servers such as diameter/radius/NTP/syslog servers in case of failures, load-balancing or when moving devices to different locations in the network where better local server instances should be used.

[RFC8368] was written on the simple assumption that all server instances for services described in this document, NTP, Radius/Diameter, Syslog and so on are located in a so called 'Network Operations Center'. This was at the writing of [RFC8368] how this was done and called in various, mostly Enterprise networks, but is today not necessarily a good way to capture all possible deployment options. For example, server instances can today with distributed Point of Presence (POP) and edge data-centers much easier decentralized for resilience, performance and cost. Therefore, this document avoids limiting its applicability to just one "NOC" deployment option.

## 1.2. ACP nodes supporting services autoconfiguration

This document introduces the term ACPna nodes to indicate nodes supporting ACP that also support the requirements described in this document: ACP (n)oc (a)utoconfigurable.

If an ACPna node supports zero-touch bootstrap of the ACP where no configuration is possible before the ACP is enabled, then the services autoconfiguration features described in this document SHOULD be enabled by default on such an ACPna node after this zero-touch bootstrap, because the autoconfiguration of these services can be the

only method for the ACPna node to become operationally accessible from OAM systems so that it can further be configured. ANI nodes are nodes supporting ACP and BRSKI ([RFC8995]). BRSKI bootstrap is an instance of such a zero-touch bootstrap requiring auto-enablement of autoconfiguration after zero-toch bootstrap.

If an ACPna node was not zero-touch bootstrapped, then autoconfiguration SHOULD be enabled whenever ACP is enabled but may be separately configurable.

### 1.3. Use of ACP GRASP for autoconfiguration

Autoconfiguration of ACNna services utilizes the ACP instance of GRASP, ([RFC8990] as defined in [RFC8994]). It leverages and extends the GRASP objective definitions of [I-D.eckert-anima-grasp-dnssd]. Thos objective elements allow to create DNS-SD compatible service announcements with flexible priority/weight and distance based selection across multiple service instances and per-service parameters.

Nodes supporting a particular service announce it via the appropriate GRASP objective into ACP GRASP. The nodes therefore need to have access to the ACP, either directly because they are ACP nodes or because they use the ACP connect function (see [RFC8994]). ACPna nodes receive these announcements and auto-configure the services tied to them. In most instances, the service announcement is from server that a client instance on the ACPna node connects to, for example a syslog server in the POP/NOC or other location with compute. In another instance, the service is that of an authentication service and the ACPna nodes will enable a server instance that leverages the authentication service elsewhere in the network.

Note: Currently, this document does not define the option of an mDNS/DNS-SD -> ACP GRASP gateway function to enable service nodes without GRASP implementations to utilize mDNS/DNS-SD to announce their services and then expect an appropriate translation function to convert these announcements into GRASP objectives. This document does define all the GRASP objectives so that that it would be possible to define such a gateway function, but some loss of functionality would exist. For once, GRASP does support network distance based service selection (e.g., select a server from the closest service node location), whereas no such mechanism exists in DNS-SD. In addition, this documen believes that support of GRASP software to announce services from service systems is very easy to accomplish.

#### 1.4. GRASP parameters

Unless otherwise described, all GRASP objective announcements described in this document SHOULD default to the following GRASP parameters. These parameters MAY all be configurable on the service nodes.

- \* M\_FLOOD GRASP message, periodically sent once every 60 second. Random phase vs. full minutes (so different service announcements are distributed over time in the network).
- \* ttl of 210000 msec (3.5 times 60 seconds).
- \* locator-option is the ACP address of the announcing node unless the announcement is done from a third-party, for example if the announcing server does not support GRASP but GRASP is run on another service node.
- \* objective-name is 'SRV.<name>', where <name> is an [RFC6335] registered service name for the service in question.
- \* objective-flags is sync-only, loop-count is 255.
- \* objective-value MUST comply with the requirements of [I-D.eckert-anima-grasp-dnssd].

```
[M_FLOOD, 12340815, h'fd89b714f3db0000200000064000001', 210000,
  ["SRV.syslog", 4, 255,
    { rfcXXXX: {
      &(sender-loop-count:1) => 255,
      &(srv-element:2) => {
        &(msg-type:1) => &(describe: 0),
        &(service:2)  => "syslog",
        &(instance:3) => "east-coast-primary",
        &(priority:5) => 0,
        &(weight:6)   => 65535,
        &(kvpairs:7)  => { "replicate" => 2 },
        &(range:8)   => 2,
      }
    }
  ],
  [O_IPv6_LOCATOR,
    h'fd89b714f3db0000200000064000001', TLS12, 514]
]
```

Figure 1: SRV.syslog example

The above example shows the default values for a "syslog" service announcement using the objective-value elements defined in [I-D.eckert-anima-grasp-dnssd]. SRV.syslog is the standard objective name for the "syslog" service, as is SRV.<any> for the <any> service. The announcer of this objective also provides the syslog service as it is announcing its own address in the locator option. It provides syslog on the standard syslog TCP port 514 using TLS12.

The DNS-SD equivalent service attributes are carried in the srv-element. The msg-type indicates that this objective is a service announcement. The instance of "" indicates that this service announcement is for the ACP itself, and not for e.g. the data-plane. It is shown here just for illustration purposes and can be left out in encoding because it is the default. Likewise, the service element is redundant and shown only for illustrative purposes. Priority and weight have the same semantic as in DNS-SD SRV records. In this case, the service announcement indicates the highest priority (0) and the highest weight (65535). Kvpairs includes service specific options.

Going beyond the capabilities, the range parameter indicates that the client of this service should select this announced service not only by priority/weight but primarily by the distance in terms of network hop-count between this service announcer and the client: The client is expected to select the best service announcement by priority and weight only between alternatives that are not more than two network hops apart in distance to the client. Otherwise the client should pick the closer one.

To allow the client to know the distance to a service announcement, the sender-loop-count parameter is included in the announcement. It MUST be set by the sender to the same value (255 in this example) as the loop-count in the GRASP header. The loop-count in the header is hop-by-hop reduced. When the GRASP message arrives at the client, the difference between sender-loop-count and loop-count is the distance to the service announcer in hops.

```

;
; Following GRASP header definitions from GRASP
;
flood-message = [M_FLOOD, session-id, initiator, ttl,
                 +[objective, (locator-option / [])]]
objective = ["SRV.<rfc6335-name>", objective-flags, loop-count,
            objective-value]

objective-flags = sync-only ; as in GRASP spec
sync-only       = 4         ; M_FLOOD only requires synchronization
loop-count      = 255       ; recommended
;
; Following GRASP objective-value definitions from GRASP DNS-SD
;
objective-value = { 1*elements }
elements        = ( @rfcXXXX: { 1*relement } )
relement       //= ( &(sender-loop-count:1) => 1..255 )
relement       //= ( &(srv-element:2) => context-element )
context-element = {
    ?( &(private:0)      => any),
    ?( &(msg-type:1)     => msg-type),
    ?( &(service:2)      => tstr),
    *( &(instance:3)     => tstr),
    ?( &(domain:4)       => tstr),
    ?( &(priority:5)     => 0..65535 ),
    ?( &(weight:6)       => 0..65535 ),
    *( &(kvpairs:7)      => { *(tstr: any) },
    ?( &(range:8)        => 0..255 ),
    *( &(clocator:9)     => clocator),
}
;
TLS12 = 257

```

Figure 2: GRASP service definition

The above picture shows the complete CDDL definition of a GRASP M\_FLOOD message indicating a service together with the objective-value encoding. Som of the context-element options are not used in this document (TBD - remove before going RFC).

The value 257 is defined to indicate TLS12 ([RFC5246]) to be used in the protocol field of GRASP locators to indicate that a TCP port is intended to be used with TLS version 1.2. Values 1..255 are reserved for IP protocol numbers.

## 2. Services

### 2.1. Syslog

ACPna nodes SHOULD support autoconfiguring of syslog via the SRV.syslog objective.

When an ACPna node discovers one or more SRV.syslog announcements across the ACP, it SHOULD perform syslog operations to the best available discovered server.

Local configuration of syslog on the ACPna node SHOULD have no impact on the autoconfigured syslog operations, or else, misconfiguration could cause to failure of the autoconfigured syslog operations. Instead, configured syslog operations should just operate as ships-in-the-night to the GRASP learned autoconfigured syslog operations.

Severity of syslog messages SHOULD be 5 (Notice) (see [RFC5424]), and all messages that are necessary to support normal remote operations of the node should be assigned severities higher (numerically lower) or equal to 5/Notice.

Syslog service announcements SHOULD include the instance option, indicating the unique name of the service instance described by the GRASP objective. This serves diagnostics and avoids having to identify service instances by the address(es) in the locator-options. In the example Figure 1, the instance name is "east-coast-primary".

The syslog facility value is a choice of the ACPna node, the autoconfigured syslog server must be able to deal with any syslog facility code received. If an ACPna node has no pre-established standard for the facility-code, then the value of local7 (23) MAY be used.

For resilience, it may be appropriate to receive syslog messages on more than one server. A server can indicate this via the "replicate" keyword in the GRASP objective-value kvpair element. The value of the "replicate" keyword indicates the maximum number of syslog servers that the client SHOULD autoconfigure syslog to. After selecting the best service announcement, the client looks up the value N of the "replicate" keyword of that best servers announcement and selects the best N-1 service announcements and ultimately logs to all N. ACPna nodes SHOULD support autoconfigured syslog to up to 3 servers simultaneously.

Autoconfigured syslog SHOULD support TLS1.2, TCP and UDP. Because ACP provides encryption, use of just TCP instead of TLS should be sufficient and may achieve higher performance. Use of UDP should be avoided because of the potential to loose packets and not supporting congestion control.

If a syslog server supports more than one transport option (TLS1.2, TCP, UDP), it SHOULD announce them via a single GRASP objective and list them via clocator options of the srv-element because the locator-option in the GRASP header (as shown in example Figure 1) allows only one locator-option. The order of the clocator options in the indicates the preference of the server. From this list, the client supports the first option supported also by the client and ignore the others. The context of the clocator would normally be "", indicating that the locator-option address is reachable via the ACP.

Instead of (or in addition to) using multiple clocator options, a server can also announce multiple SRV.syslog objectives, but in that case each of them would be considered to be a different service instance considered by the the client when selecting the (set of) best service instances. If a service announcement indicates via the "replicate" keyword that the client should log to three service instances, and announce three separate SRV.syslog objectives, each one with a different locator-option, then the client might select to log to all three of them - instead of - which is more likely the desired option - for the client to log to actually three different servers. Hence the use of multiple clocator options that are examined by clients only after server selection is done.

When a client uses TLS, it MUST use its ACP domain certificate for authentication. Likewise, the syslog server MUSTS use its ACP domain certificate.

Logging by default uses the ACP, in the clocator option, this is indicated via a context value of "". Servers may also indicate support for logging across the data-plane, which may provide higher performance but may fail if reachability in the data-plane does not exist, so care must be taken when announcing this option. For example, in managed MPLS/VPN networks where the ACP extends across P/PE and CE devices, the global routing table on a CE device is often not the same as that on P/PE devices, and therefore CE devices may not be able to log to "0". In this case, the syslog server should instead announce a deployment choosen name for the context, such as "VRF0". Clients would only take such a clocator into account if there is a local configuration that maps the context name to a routing table. In this example, only P/PE nodes would have this configuration, therefore allowing the CE nodes to ignore this clocator; And if this clocator was the only locator-option in the

GRASP objective, then the whole objective MUST be ignored by the client when selecting the best possible service instance. Note that for contexts other than the ACP (""), both IPv4 and IPv6 are possible, depending on what version(s) of IP are deployed in the data-plane.

Failure to connect to a chosen service instance SHOULD be taken into accounts by clients when selecting service instances to log to. For UDP locator-options, ICMP/ICMPv6 error indications are such connection failures. For TCP/TLS connections, connection failure includes TCP and TLS failures as well as keepalive failures. When failures occur, clients should attempt to re-connect with exponential timeouts, starting with 5 seconds and staying at 320 seconds or until the GRASP service announcement expires and is not refreshed.

When connecting to a server fails, the ACPna client SHOULD connect to the next best available server in the meantime. ACPna client SHOULD support connecting to up to four service instances if any connections fail. If for example the client is logging to two service instances because 2 is indicated in the "replicate" option of the service announcements, and one fails, the client will attempt to re-connect to it while in parallel establishing syslog connection to a third-best service-instance.

When establishing connection to a new syslog service instance, ACPna clients SHOULD log with severity 5 an indication of this event, indicating its own ACP address, the ACP address and if existing instance name of the new syslog service instance and the reason. Like any other autoconfigured syslog message, this would go to all syslog connections and therefore show up on the redundant syslog servers, allowing to recognize failure of connectivity to another syslog server - and tracing of client logs across syslog servers if the client changes them.

Examples:

```
ACP: fd89:b714:f3db::0200:0000:6400:0042 start logging to:
fd89:b714:f3db::0200:0000:6400:0001/east-coast-primary,TLS reason:
starting up
```

```
ACP: fd89:b714:f3db::0200:0000:6400:0042 start logging to:
fd89:b714:f3db::0200:0000:6400:0001/east-coast-primary,TLS reason:
new better service instance
```

```
ACP: fd89:b714:f3db::0200:0000:6400:0042 stop logging to:
fd89:b714:f3db::0200:0000:6400:0001/east-coast-secondary,TLS reason:
connection failure
```

When failures to deliver syslog messages to ANY syslog servers happen, clients SHOULD track the this and indicate loss of messages via the next working syslog connection. Note that due to the possibility of ICMP/ICMPv6 errors, only the successful delivery of messages via TLS or TCP should be tracked. TBD: need to check if this can reasonably be recommended, pending on availability of e.g. TAPS API spec to know whether a TCP write was sent and acknowledged by the receiver (given how there are no reply messages in syslog).

## 2.2. NTP

Time synchronization is one of the most fundamental functionality for network devices for a variety of functions to work and also for diagnostics to be comparable across the network. If problems propagate fast across the network, the client generated timestamp of events in syslog messages (or other diagnostics function) allows to trace event propagation and deduce causality. This may require network clock synchronization in the order of milliseconds, something which is easily achievable in today's network devices via NTP.

ACPna nodes SHOULD support autoconfiguration of clock synchronization through NTP ([RFC5905]) with the following autoconfiguration semantics.

The GRASP objective for NTP is SRV.ntp. This does not distinguish between NTPv4 and NTPv3 because NTPv4 is fully backward compatible with NTPv3, so server and client will negotiate between these two versions.

The kvpair key "stratum" has a numeric value and indicates the stratum or level of a server in a synchronization tree. The value of 1 indicates the root of the distribution tree. Servers that synchronize from the master have a stratum of 2, and so on.

The kvpair key "minpoll" indicates the lowest periodic polling that the client will perform against the server. Announcing a large numeric value allows for a server to reduce the amount of NTP messages from clients, but slows down convergence time of clients number of service instances that simultaneously bootstrap.

The kvpair key "key" indicates the NTPv3 authentication mechanism. When present, clients MUST use the value as the key to perform NTPv3 (MD5) hash authentication of message with this service instance. Note that the encryption of the ACP serves as protection of distributing such a cleartext symmetric key via GRASP to clients.

TBD: Understand NTPv4 autokey and define appropriate kvpair to enable auto-configuring it, especially when the service instance announcement indicates the use of the data-plane.

The autoconfiguration described in the following paragraphs is for leafs of the clock distribution graph, e.g., nodes that do only aim to obtain synchronized time from a server. Configuration of the server hierarchy is left to explicit configuration.

Clients SHOULD select service instance(s) with the worst (highest) stratum value. In the face of multiple equal options, clients have to pick the best ones based on the standard selection criteria priority/weight and range, allowing for distributed NTP server deployment by e.e., setting range to 1, or via centralized deployment with multiple servers, setting range to 255 and priority/weight accordingly. Making the stratum the primary selection criteria allows in the future to also introduce autoconfiguration of servers in the NTP clock distribution tree without incurring the problem that a large number of clients would then select higher stratum servers (and overload them).

Like most other autoconfigured services, the autoconfigured NTP time synchronization SHOULD take precedence over explicit configured NTP options to ensure that time synchronization is not subject to misconfiguration of individual nodes (but only subject to misconfiguration of servers).

The kvpair "TZ" option allows to signal the time zone of the ACP network to clients. Its value is a string indicating the time zone of all nodes in the ACP network. Care must be taken not to use this option in networks extending across multiple time zones. Because time zone distribution does not work automatically across larger networks with multiple time zones, overriding the signalled time zone SHOULD be possible through local configuration.

TBD: references for time zone spec standards and also for DST rule indications.

### 2.3. DNS for operations

Availability of DNS names for network operations/troubleshooting is today mostly an convenience in network operations, but with IPv6 evolving the need to use DNS names even in CLI based network diagnostics is raising - because IPv6 addresses often are more difficult to memorize by operators. More and more network features also support configurition that instead of addresses include domain names or URLs, and ultimately, any non-fully autoconfigured functions should rather rely on domain-names and URLs instead of just addresses

for greater flexibility and reliability in the face of address changes.

In the face of this, ACPna nodes SHOULD support autoconfiguration of DNS for operational purposes. "For operational purposes" implies that the use of the autoconfigured DNS servers SHOULD NOT be used for DNS services offered to users of the data plane, such as DNS proxy services. This would cause the ACP to effectively carry user traffic, whenever a client DNS request to an ACPna node with a DNS proxy would be forwarded to an autoconfigured server via the ACP.

The GRASP objective name for such OAM use of DNS is OAM-DNS. It is explicitly not SRV.dns to highlight that this instance of DNS is copied for operational purposes only to isolate it from user issues (performance across the ACP and attacks). Utilizing different DNS contexts also allows to set up split-horizon DNS where all the operationally relevant DNS names are only made available via the DNS servers or zones available across the ACP.

The value of the "search-list" kvpair option is a ";" (semicolon) separated list of domain name prefixes that should be searched by the client for non-FQDN that they need to resolve. "local-arpa" is the prefix to use for reverse IPv4/IPv6 address lookups. If for example "local-arpa" is set to "arpa.example.com", then the clients should first look up IPv4/IPv6 addresses in "ipv6.arpa.example.com"/"in-addr.arpa.example.com." before resorting to lookup in the Internet global "ipv6.arpa"/"in-addr.arpa.". For RFC1918/ULA addresses, no fallback to the global reverse lookup prefixes should be done.

ACPna nodes SHOULD look up their name via a reverse lookup of their ACP address, and then auto-configure this name.

There are no service specifics for the selection of DNS servers. A ACPna node simply uses the standard priority/weight/range options to select a DNS server. It MAY prefer a server with TCP locator-option simply because that allows in most cases faster discovery of connectivity problems than a UDP connection.

TBD: Note that it is fairly easy to re-use the autoconfiguration scheme described here to provide auto-configuration of DNS for user DNS services with the help of the ACP. The objective name would have to be changed and the locators would have to indicate a data-plane context, so that user requests are carried across the data-plane from DNS proxies to DNS servers. It is unclear if this service should be described in this document though.

## 2.4. Radius

Radius [RFC2865] is a protocol used for AAA service - Authentication, Authorization and Accounting. Autodiscovery of Radius servers across the ACP for ACPna nodes serves the purpose to enable authentication and authorization of other ACPna autoconfigured services such as below described Section 2.6.

ACPna nodes MUST support Radius and/or Diameter autoconfiguration if they support any of the autoconfigured services depending on such an authentication service.

The GRASP objective name is SRV.radius. The UDP or TCP port of the locator-option in the GRASP header or the clocator option indicate the UDP or TCP port of the Radius servers authentication connection. The context of a clocator MUST be "" to indicate the ACP - because the Radius connections MUST pass across the ACP to be protected against eavesdropping - and the radius security methods described here are not sufficiently secure to allow passing them across the data-plane.

The kvpair "secret\_key" string value indicates the secret key to use on the connection to the Radius server. The optional "acct\_port" numeric value indicate the UDP/TCP port of any accounting connection supported by the radius server. The protocol (UDP vs. TCP) is the same as the one in the chosen locator-option/clocator.

There are no service specific selection rules. TCP is preferred for faster recognition of a failed server and reselection of an alternative server.

The specific data/authentication/authorization configuration required on the Radius server depends on the OAM service authenticated/authorized and is described in its section in this document.

TBD: Should we define AVpair or different objective names to distinguish what services can be authenticated ? Would be easier if we found another service than SSH/Netconf.

## 2.5. Diameter

TBD. Alternative to Radius. Author would welcome suggesting what parameters are relevant for a diameter authentication service.

## 2.6. SSH server

ACPna nodes supporting SSH server functionality for remote management access via CLI, NETCONF or other methods SHOULD auto-enable SSH server functionality across the ACP whenever they are aware from ACP GRASP of RADIUS (Section 2.4) or DIAMETER (Section 2.5) authentication servers. ACPna nodes that support ACPna SSH server functionality MUST support authentication via either RADIUS and/or Diameter.

If both protocols are supported by the ACPna node, the ACPna node SHOULD select the authentication server based on the service priority parameters across both protocols. E.g., if a RADIUS server has a higher priority in GRASP than the DIAMETER server, the ACPna node should authenticate against the RADIUS server.

When valid authentication server(s) are discovered, the SSH server is autoconfigured. It SHOULD only listen to the standard SSH port with the ACP address of the node but not be reachable from the data-plane. It MUST NOT be modifyable by configuration (only by auto-configuration). If autoconfiguration of an SSH server on the standard SSH port conflicts with explicitly configured SSH server for the data-plane due to software limitations or complexity, the autoconfigured SSH server MAY be started on a node-type specific and not dynamically selected port number. This port number must be well-known to OAM operations as there is no method provided to signal it to the SSH client side.

Note that this document does not define any standards for the exact message options for authentication or authorization. Especially authorization, such as privilege level that permits to change configuration is likely using vendor specific methods, and Radius/Diameter servers must be capable to recognize the type of client as they had to without this autoconfiguration.

## 3. Security Considerations

There is no protection against "unauthorized" ACP nodes to generate service announcements, because there is no authorization scheme in GRASP. Discovery of unauthorized announcers is easy though because the service announcements are flooded across the ACP and are therefore easily visible on nodes that may specifically observe announcements to discover unauthorized ones.

A possible framework to define authorization could rely on defining roles for ACP nodes either through additional parameters in their ACP domain certificate or following initial provisioning, and then lock down the ability for later configuration to enable services (and their GRASP announcements) to only those included in the role assigned to the node. This is outside the scope of this document.

#### 4. Acknowledgments

Thanks to Ignas Bagdonas for deployment / applicability / terminology input and to Balaji BL, Ravi Kumar Vadapalli for their original implementation of the concept.

#### 5. Change log [RFC Editor: Please remove]

draft-eckert-anima-services-dns-autoconfig

01: Refresh.

00: Renaming from 'noc-autoconfig' after a long discussion with Ignas Bagdonas: replaced all mention of NOC with "infrastructure / decentralized" services, because the term NOC seems to be a terminology that does not well match how it is called in many type of networks.

draft-eckert-anima-noc-autoconfig (2018)

00: Initial version.

#### 6. References

##### 6.1. Normative References

[I-D.eckert-anima-grasp-dnssd]

Eckert, T., Boucadair, M., Jacquenet, C., and M. H. Behringer, "DNS-SD Compatible Service Discovery in GeneRIC Autonomic Signaling Protocol (GRASP)", Work in Progress, Internet-Draft, draft-eckert-anima-grasp-dnssd-02, 12 July 2021, <<https://www.ietf.org/archive/id/draft-eckert-anima-grasp-dnssd-02.txt>>.

[RFC2865]

Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <<https://www.rfc-editor.org/info/rfc2865>>.

- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <<https://www.rfc-editor.org/info/rfc5424>>.
- [RFC5905] Mills, D., Martin, J., Ed., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, DOI 10.17487/RFC5905, June 2010, <<https://www.rfc-editor.org/info/rfc5905>>.
- [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <<https://www.rfc-editor.org/info/rfc6335>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC8368] Eckert, T., Ed. and M. Behringer, "Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM)", RFC 8368, DOI 10.17487/RFC8368, May 2018, <<https://www.rfc-editor.org/info/rfc8368>>.
- [RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <<https://www.rfc-editor.org/info/rfc8990>>.
- [RFC8994] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "An Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <<https://www.rfc-editor.org/info/rfc8994>>.

## 6.2. Informative References

- [RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <<https://www.rfc-editor.org/info/rfc8995>>.

Authors' Addresses

Toerless Eckert  
Futurewei Technologies USA Inc.  
2220 Central Expressway  
Santa Clara, 95050  
United States of America  
Email: tte+ietf@cs.fau.de

Mohamed Boucadair  
Orange  
35000 Rennes  
France  
Email: mohamed.boucadair@orange.com

Christian Jacquenet  
Orange  
Email: christian.jacquenet@orange.com

Michael H. Behringer  
Email: michael.h.behringer@gmail.com

ANIMA WG  
Internet-Draft  
Intended status: Standards Track  
Expires: 8 September 2022

D. von Oheimb, Ed.  
S. Fries  
H. Brockhaus  
Siemens  
E. Lear  
Cisco Systems  
7 March 2022

BRSKI-AE: Alternative Enrollment Protocols in BRSKI  
draft-ietf-anima-brski-async-enroll-05

Abstract

This document enhances Bootstrapping Remote Secure Key Infrastructure (BRSKI, [RFC8995]) to allow employing alternative enrollment protocols, such as CMP.

Using self-contained signed objects, the origin of enrollment requests and responses can be authenticated independently of message transfer. This supports end-to-end security and asynchronous operation of certificate enrollment and provides flexibility where to authenticate and authorize certification requests.

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 8 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
1.1. Motivation . . . . .	3
1.2. Supported environment . . . . .	5
1.3. List of application examples . . . . .	6
2. Terminology . . . . .	6
3. Requirements discussion and mapping to solution elements . .	7
4. Adaptations to BRSKI . . . . .	10
4.1. Architecture . . . . .	10
4.2. Message exchange . . . . .	13
4.2.1. Pledge - Registrar discovery and voucher exchange . .	13
4.2.2. Registrar - MASA voucher exchange . . . . .	13
4.2.3. Pledge - Registrar - RA/CA certificate enrollment . .	13
4.2.4. Pledge - Registrar - enrollment status telemetry . .	16
4.2.5. Addressing scheme enhancements . . . . .	16
4.3. Domain registrar support of alternative enrollment protocols . . . . .	16
5. Examples for signature-wrapping using existing enrollment protocols . . . . .	17
5.1. Instantiation to EST (informative) . . . . .	17
5.2. Instantiation to CMP (normative if CMP is chosen) . . . .	18
6. IANA Considerations . . . . .	18
7. Security Considerations . . . . .	19
8. Acknowledgments . . . . .	19
9. References . . . . .	19
9.1. Normative References . . . . .	19
9.2. Informative References . . . . .	20
Appendix A. Using EST for certificate enrollment . . . . .	21
Appendix B. Application examples . . . . .	22
B.1. Rolling stock . . . . .	23
B.2. Building automation . . . . .	23
B.3. Substation automation . . . . .	24
B.4. Electric vehicle charging infrastructure . . . . .	24
B.5. Infrastructure isolation policy . . . . .	24
B.6. Sites with insufficient level of operational security . .	25
Appendix C. History of changes TBD RFC Editor: please delete . .	25
Authors' Addresses . . . . .	29

## 1. Introduction

### 1.1. Motivation

BRSKI, as defined in [RFC8995], specifies a solution for secure automated zero-touch bootstrapping of new devices, so-called pledges. This includes the discovery of the registrar in the target domain, time synchronization, and the exchange of security information necessary to establish mutual trust between pledges and the target domain.

A pledge gains trust in the target domain via the domain registrar as follows. It obtains security information about the domain, specifically a domain certificate to be trusted, by requesting a voucher object defined in [RFC8366]. Such a voucher is a self-contained signed object originating from a Manufacturer Authorized Signing Authority (MASA). Therefore, the voucher may be provided in online mode (synchronously) or offline mode (asynchronously). The pledge can authenticate the voucher because it is shipped with a trust anchor of its manufacturer such that it can validate signatures (including related certificates) by the MASA.

Trust by the target domain in a pledge is established by providing the pledge with a domain-specific LDevID certificate. The certification request of the pledge is signed using its IDevID secret and can be validated by the target domain using the trust anchor of the pledge manufacturer, which needs to be pre-installed in the domain.

For enrolling devices with LDevID certificates, BRSKI typically utilizes Enrollment over Secure Transport (EST) [RFC7030]. EST has its specific characteristics, detailed in Appendix A. In particular, it requires online or on-site availability of the RA for performing the data origin authentication and final authorization decision on the certification request. This type of enrollment can be called 'synchronous enrollment'. For various reasons, it may be preferable to use alternative enrollment protocols such as the Certificate Management Protocol (CMP) [RFC4210] profiled in [I-D.ietf-lamps-lightweight-cmp-profile] or Certificate Management over CMS (CMC) [RFC5272], that are more flexible and independent of the transfer mechanism because they represent certification request messages as authenticated self-contained objects.

Depending on the application scenario, the required RA/CA components may not be part of the registrar. They even may not be available on-site but rather be provided by remote backend systems. The registrar or its deployment site may not have an online connection with them or the connectivity may be intermittent. This may be due to security requirements for operating the backend systems or due to site

deployments where on-site or always-online operation may be not feasible or too costly. In such scenarios, the authentication and authorization of certification requests will not or can not be performed on-site at enrollment time. In this document, enrollment that is not performed in a (time-wise) consistent way is called \_asynchronous enrollment\_. Asynchronous enrollment requires a store-and-forward transfer of certification requests along with the information needed for authenticating the requester. This allows offline processing the request.

Application scenarios may also involve network segmentation, which is utilized in industrial systems to separate domains with different security needs. Such scenarios lead to similar requirements if the TLS connection carrying the requester authentication is terminated and thus request messages need to be forwarded on further channels before the registrar/RA can authorize the certification request. In order to preserve the requester authentication, authentication information needs to be retained and ideally bound directly to the certification request.

There are basically two approaches for forwarding certification requests along with requester authentication information:

- \* A trusted component (e.g., a local RA) in the target domain is needed that forwards the certification request combined with the validated identity of the requester (e.g., its IDevID certificate) and an indication of successful verification of the proof-of-possession (of the corresponding private key) in a way preventing changes to the combined information. When connectivity is available, the trusted component forwards the certification request together with the requester information (authentication and proof-of-possession) for further processing. This approach offers only hop-by-hop security. The backend PKI must rely on the local pledge authentication result provided by the local RA when performing the authorization of the certification request. In BRSKI, the EST server is such a trusted component, being co-located with the registrar in the target domain.
- \* Involved components use authenticated self-contained objects for the enrollment, directly binding the certification request and the requester authentication in a cryptographic way. This approach supports end-to-end security, without the need to trust in intermediate domain components. Manipulation of the request and the requester identity information can be detected during the validation of the self-contained signed object.

Focus of this document is the support of alternative enrollment protocols that allow using authenticated self-contained objects for device credential bootstrapping. This enhancement of BRSKI is named BRSKI-AE, where AE stands for alternative enrollment protocols and for asynchronous enrollment. This specification carries over the main characteristics of BRSKI, namely that the pledge obtains trust anchor information for authenticating the domain registrar and other target domain components as well as a domain-specific X.509 device certificate (the LDevID certificate) along with the corresponding private key (the LDevID secret) and certificate chain.

The goals are to enhance BRSKI to

- \* support alternative enrollment protocols,
- \* support end-to-end security for enrollment, and
- \* make it applicable to scenarios involving asynchronous enrollment.

This is achieved by

- \* extending the well-known URI approach with an additional path element indicating the enrollment protocol being used, and
- \* defining a certificate waiting indication and handling, for the case that the certifying component is (temporarily) not available.

This specification can be applied to both synchronous and asynchronous enrollment.

In contrast to BRSKI, this specification supports offering multiple enrollment protocols on the infrastructure side, which enables pledges and their developers to pick the preferred one.

## 1.2. Supported environment

BRSKI-AE is intended to be used in domains that may have limited support of on-site PKI services and comprises application scenarios like the following.

- \* There are requirements or implementation restrictions that do not allow using EST for enrolling an LDevID certificate.
- \* Pledges and/or the target domain already have an established certificate management approach different from EST that shall be reused (e.g., in brownfield installations).

- \* There is no registration authority available on site in the target domain. Connectivity to an off-site RA is intermittent or entirely offline. A store-and-forward mechanism is used for communicating with the off-site services.
- \* Authoritative actions of a local RA are limited and may not be sufficient for authorizing certification requests by pledges. Final authorization is done by an RA residing in the operator domain.

### 1.3. List of application examples

Bootstrapping can be handled in various ways, depending on the application domains. The informative Appendix B provides illustrative examples from various industrial control system environments and operational setups. They motivate the support of alternative enrollment protocols, based on the following examples of operational environments:

- \* Rolling stock
- \* Building automation
- \* Electrical substation automation
- \* Electric vehicle charging infrastructures
- \* Infrastructure isolation policy
- \* Sites with insufficient level of operational security

## 2. Terminology

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

This document relies on the terminology defined in [RFC8995] and [IEEE.802.1AR\_2009]. The following terms are defined in addition:

EE: End entity, in the BRSKI context called pledge. It is the entity that is bootstrapped to the target domain. It holds a public-private key pair, for which it requests a public-key certificate. An identifier for the EE is given as the subject name of the certificate.

RA: Registration authority, an optional system component to which a CA delegates certificate management functions such as authenticating requesters and performing authorization checks on certification requests.

CA: Certification authority, issues certificates and provides certificate status information.

target domain: The set of entities that share a common local trust anchor, independent of where the entities are deployed.

site: Describes the locality where an entity, e.g., pledge, registrar, RA, CA, is deployed. Different sites can belong to the same target domain.

on-site: Describes a component or service or functionality available in the target deployment site.

off-site: Describes a component or service or functionality available in an operator site different from the target deployment site. This may be a central site or a cloud service, to which only a temporary connection is available.

asynchronous communication: Describes a time-wise interrupted communication between a pledge (EE) and a registrar or PKI component.

synchronous communication: Describes a time-wise uninterrupted communication between a pledge (EE) and a registrar or PKI component.

authenticated self-contained object: Describes in this context an object that is cryptographically bound to the IDevID certificate of a pledge. The binding is assumed to be provided through a digital signature of the actual object using the IDevID secret.

### 3. Requirements discussion and mapping to solution elements

There were two main drivers for the definition of BRSKI-AE:

- \* The solution architecture may already use or require a certificate management protocol other than EST. Therefore, this other protocol should be usable for requesting LDevID certificates.
- \* The domain registrar may not be the (final) point that authenticates and authorizes certification requests and the pledge may not have a direct connection to it. Therefore, certification requests should be self-contained signed objects.

Based on the intended target environment described in Section 1.2 and the application examples described in Appendix B, the following requirements are derived to support authenticated self-contained objects as containers carrying certification requests.

At least the following properties are required:

- \* **proof-of-possession:** demonstrates access to the private key corresponding to the public key contained in a certification request. This is typically achieved by a self-signature using the corresponding private key.
- \* **proof-of-identity:** provides data origin authentication of the certification request. This typically is achieved by a signature using the IDevID secret of the pledge.

Here is an incomplete list of solution examples, based on existing technology described in IETF documents:

- \* **Certification request objects:** Certification requests are data structures protecting only the integrity of the contained data and providing proof-of-possession for a (locally generated) private key. Examples for certification request data structures are:
  - **PKCS#10 [RFC2986].** This certification request structure is self-signed to protect its integrity and prove possession of the private key that corresponds to the public key included in the request.
  - **CRMF [RFC4211].** Also this certificate request message format supports integrity protection and proof-of-possession, typically by a self-signature generated over (part of) the structure with the private key corresponding to the included public key. CRMF also supports further proof-of-possession methods for types of keys that do not support any signature algorithm.

The integrity protection of certification request fields includes the public key because it is part of the data signed by the corresponding private key. Yet note that for the above examples this is not sufficient to provide data origin authentication, i.e., proof-of-identity. This extra property can be achieved by an additional binding to the IDevID of the pledge. This binding to source authentication supports the authorization decision for the certification request. The binding of data origin authentication to the certification request may be delegated to the protocol used for certificate management.

- \* Solution options for proof-of-identity: The certification request should be bound to an existing authenticated credential (here, the IDevID certificate) to enable a proof of identity and, based on it, an authorization of the certification request. The binding may be achieved through security options in an underlying transport protocol such as TLS if the authorization of the certification request is (completely) done at the next communication hop. This binding can also be done in a transport-independent way by wrapping the certification request with signature employing an existing IDevID. In the BRSKI context, this will be the IDevID. This requirement is addressed by existing enrollment protocols in various ways, such as:
  - EST [RFC7030] utilizes PKCS#10 to encode the certification request. The Certificate Signing Request (CSR) optionally provides a binding to the underlying TLS session by including the tls-unique value in the self-signed PKCS#10 structure. The tls-unique value results from the TLS handshake. Since the TLS handshake includes client authentication and the pledge utilizes its IDevID for it, the proof-of-identity is provided by such a binding to the TLS session. This can be supported using the EST /simpleenroll endpoint. Note that the binding of the TLS handshake to the CSR is optional in EST. As an alternative to binding to the underlying TLS authentication in the transport layer, [RFC7030] sketches wrapping the CSR with a Full PKI Request message using an existing certificate.
  - SCEP [RFC8894] supports using a shared secret (passphrase) or an existing certificate to protect CSRs based on SCEP Secure Message Objects using CMS wrapping ([RFC5652]). Note that the wrapping using an existing IDevID in SCEP is referred to as renewal. Thus SCEP does not rely on the security of the underlying transfer.
  - CMP [RFC4210] supports using a shared secret (passphrase) or an existing certificate, which may be an IDevID credential, to authenticate certification requests via the PKIProtection structure in a PKIMessage. The certification request is typically encoded utilizing CRMF, while PKCS#10 is supported as an alternative. Thus CMP does not rely on the security of the underlying transfer protocol.

- CMC [RFC5272] also supports utilizing a shared secret (passphrase) or an existing certificate to protect certification requests, which can be either in CRMF or PKCS#10 structure. The proof-of-identity can be provided as part of a FullCMCRequest, based on CMS [RFC5652] and signed with an existing IDevID secret. Thus CMC does not rely on the security of the underlying transfer protocol.

#### 4. Adaptations to BRSKI

In order to support alternative enrollment protocols, asynchronous enrollment, and more general system architectures, BRSKI-AE lifts some restrictions of BRSKI [RFC8995]. This way, authenticated self-contained objects such as those described in Section 3 above can be used for certificate enrollment.

The enhancements needed are kept to a minimum in order to ensure reuse of already defined architecture elements and interactions. In general, the communication follows the BRSKI model and utilizes the existing BRSKI architecture elements. In particular, the pledge initiates communication with the domain registrar and interacts with the MASA as usual.

##### 4.1. Architecture

The key element of BRSKI-AE is that the authorization of a certification request **MUST** be performed based on an authenticated self-contained object. The certification request is bound in a self-contained way to a proof-of-origin based on the IDevID. Consequently, the authentication and authorization of the certification request **MAY** be done by the domain registrar and/or by other domain components. These components may be offline or reside in some central backend of the domain operator (off-site) as described in Section 1.2. The registrar and other on-site domain components may have no or only temporary (intermittent) connectivity to them. The certification request **MAY** also be piggybacked on another protocol.

This leads to generalizations in the placement and enhancements of the logical elements as shown in Figure 1.

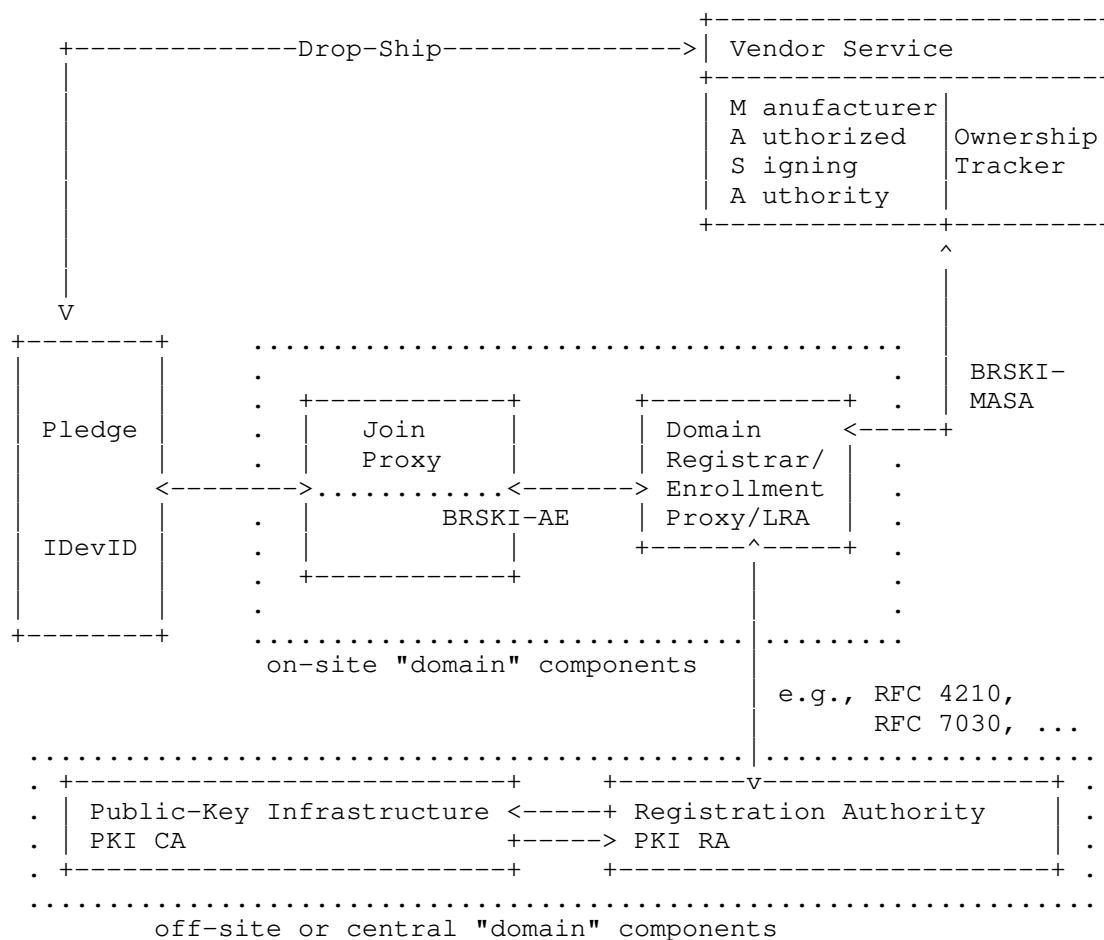


Figure 1: Architecture overview using off-site PKI components

The architecture overview in Figure 1 has the same logical elements as BRSKI, but with more flexible placement of the authentication and authorization checks on certification requests. Depending on the application scenario, the registrar MAY still do all of these checks (as is the case in BRSKI), or part of them, or none of them.

The following list describes the on-site components in the target domain of the pledge shown in Figure 1.

\* Join Proxy: same functionality as described in BRSKI [RFC8995].

- \* Domain Registrar / Enrollment Proxy / LRA: in BRSKI-AE, the domain registrar has mostly the same functionality as in BRSKI, namely to facilitate the communication of the pledge with the MASA and the PKI. Yet in contrast to BRSKI, the registrar offers different enrollment protocols and MAY act as a local registration authority (LRA) or simply as an enrollment proxy. In such cases, the domain registrar forwards the certification request to some off-site RA component, which performs at least part of the authorization. This also covers the case that the registrar has only intermittent connection and forwards the certification request to the RA upon re-established connectivity.

Note: To support alternative enrollment protocols, the URI scheme for addressing the domain registrar is generalized (see Section 4.2.5).

The following list describes the components provided by the vendor or manufacturer outside the target domain.

- \* MASA: general functionality as described in BRSKI [RFC8995]. The voucher exchange with the MASA via the domain registrar is performed as described in BRSKI.

Note: The interaction with the MASA may be synchronous (voucher request with nonce) or asynchronous (voucher request without nonce).

- \* Ownership tracker: as defined in BRSKI.

The following list describes the target domain components that can optionally be operated in the off-site backend of the target domain.

- \* PKI RA: Performs certificate management functions for the domain as a centralized public-key infrastructure for the domain operator. As far as not already done by the domain registrar, it performs the final validation and authorization of certification requests.
- \* PKI CA: Performs certificate generation by signing the certificate structure requested in already authenticated and authorized certification requests.

Based on the diagram in Section 2.1 of BRSKI [RFC8995] and the architectural changes, the original protocol flow is divided into three phases showing commonalities and differences to the original approach as follows.

- \* Discovery phase: same as in BRSKI steps (1) and (2)

- \* Voucher exchange phase: same as in BRSKI steps (3) and (4).
- \* Enrollment phase: step (5) is changed to employing an alternative enrollment protocol that uses authenticated self-contained objects.

#### 4.2. Message exchange

The behavior of a pledge described in Section 2.1 of BRSKI [RFC8995] is kept with one exception. After finishing the Imprint step (4), the Enroll step (5) MUST be performed with an enrollment protocol utilizing authenticated self-contained objects. Section 5 discusses selected suitable enrollment protocols and options applicable.

##### 4.2.1. Pledge - Registrar discovery and voucher exchange

The discovery phase and voucher exchange are applied as specified in [RFC8995].

##### 4.2.2. Registrar - MASA voucher exchange

This voucher exchange is performed as specified in [RFC8995].

##### 4.2.3. Pledge - Registrar - RA/CA certificate enrollment

As stated in Section 3, the enrollment MUST be performed using an authenticated self-contained object providing not only proof-of-possession but also proof-of-identity (source authentication).

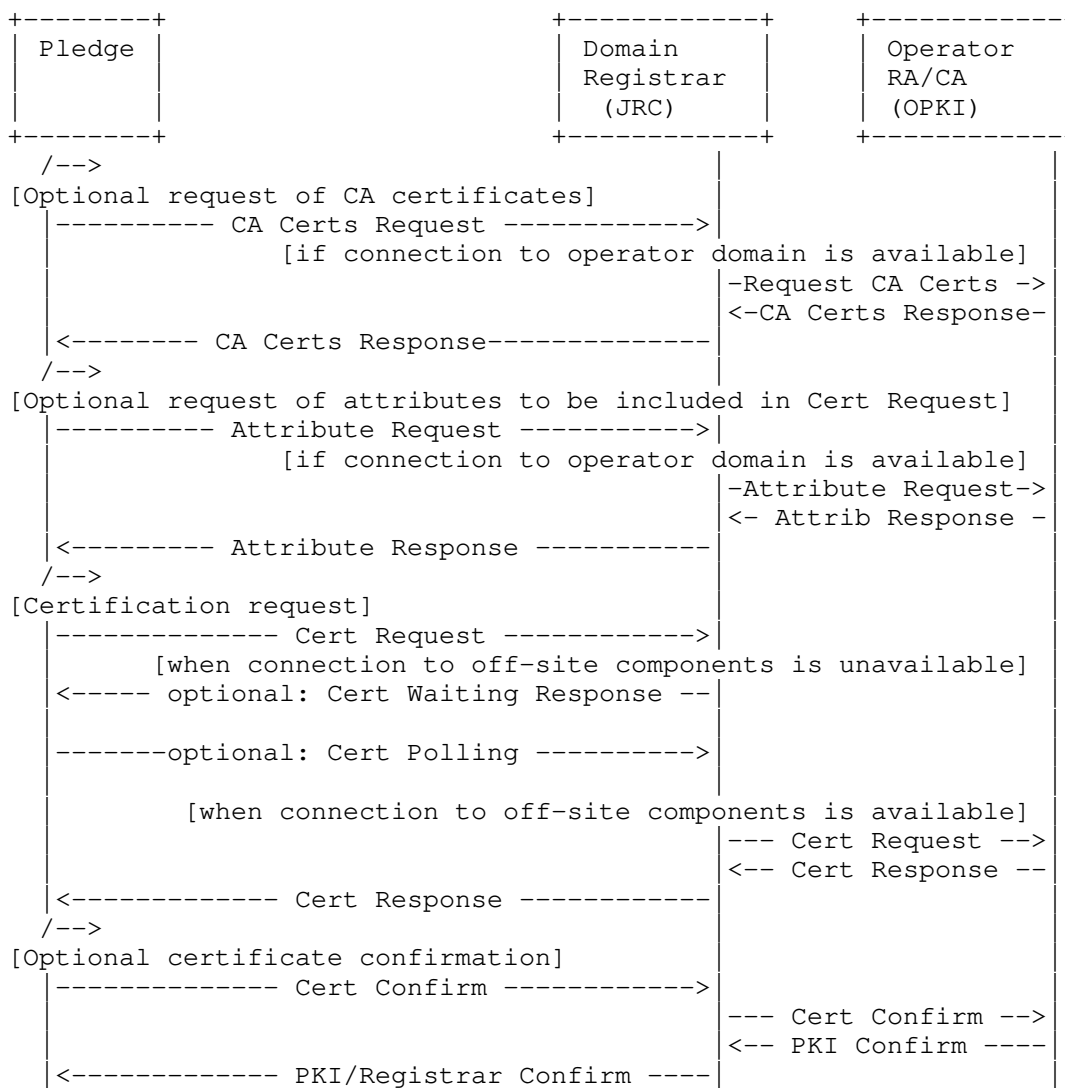


Figure 2: Certificate enrollment

The following list provides an abstract description of the flow depicted in Figure 2.

- \* CA Cert Request: The pledge optionally requests the latest relevant CA certificates. This ensures that the pledge has the complete set of current CA certificates beyond the pinned-domain-cert (which is contained in the voucher and may be just the domain registrar certificate).

- \* CA Cert Response: It MUST contain the current root CA certificate, which typically is the LDevID trust anchor, and any additional certificates that the pledge may need to validate certificates.
- \* Attribute Request: Typically, the automated bootstrapping occurs without local administrative configuration of the pledge. Nevertheless, there are cases in which the pledge may also include additional attributes specific to the target domain into the certification request. To get these attributes in advance, the attribute request can be used.
- \* Attribute Response: It MUST contain the attributes to be included in the subsequent certification request.
- \* Cert Request: This certification request MUST contain the authenticated self-contained object ensuring both proof-of-possession of the corresponding private key and proof-of-identity of the requester.
- \* Cert Response: The certification response message MUST contain on success the requested certificate and MAY include further information, like certificates of intermediate CAs.
- \* Cert Waiting Response: Optional waiting indication for the pledge, which SHOULD poll for a Cert Response after a given time. To this end, a request identifier is necessary. The request identifier may be either part of the enrollment protocol or can be derived from the certification request.
- \* Cert Polling: This SHOULD be used by the pledge in reaction to a Cert Waiting Response to query the registrar whether the certification request meanwhile has been processed. It MUST be answered either by another Cert Waiting, or the Cert Response.
- \* Cert Confirm: An optional confirmation sent after the requested certificate has been received and validated. It contains a positive or negative confirmation by the pledge whether the certificate was successfully enrolled and fits its needs.
- \* PKI/Registrar Confirm: An acknowledgment by the PKI or registrar that MUST be sent on reception of the Cert Confirm.

The generic messages described above may be implemented using various enrollment protocols supporting authenticated self-contained objects, as described in Section 3. Examples are available in Section 5.

#### 4.2.4. Pledge - Registrar - enrollment status telemetry

The enrollment status telemetry is performed as specified in [RFC8995]. In BRSKI this is described as part of the enrollment phase, but due to the generalization on the enrollment protocol described in this document it fits better as a separate step here.

#### 4.2.5. Addressing scheme enhancements

BRSKI-AE provides generalizations to the addressing scheme defined in BRSKI [RFC8995] to accommodate alternative enrollment protocols that use authenticated self-contained objects for certification requests. As this is supported by various existing enrollment protocols, they can be directly employed (see also Section 5).

The addressing scheme in BRSKI for certification requests and the related CA certificates and CSR attributes retrieval functions uses the definition from EST [RFC7030]; here on the example of simple enrollment: `"/.well-known/est/simpleenroll"`. This approach is generalized to the following notation: `"/.well-known/<enrollment-protocol>/<request>"` in which `<enrollment-protocol>` refers to a certificate enrollment protocol. Note that enrollment is considered here a message sequence that contains at least a certification request and a certification response. The following conventions are used in order to provide maximal compatibility to BRSKI:

- \* `<enrollment-protocol>`: MUST reference the protocol being used, which MAY be CMP, CMC, SCEP, EST [RFC7030] as in BRSKI, or a newly defined approach.

Note: additional endpoints (well-known URIs) at the registrar may need to be defined by the enrollment protocol being used.

- \* `<request>`: if present, the `<request>` path component MUST describe, depending on the enrollment protocol being used, the operation requested. Enrollment protocols are expected to define their request endpoints, as done by existing protocols (see also Section 5).

#### 4.3. Domain registrar support of alternative enrollment protocols

Well-known URIs for various endpoints on the domain registrar are already defined as part of the base BRSKI specification or indirectly by EST. In addition, alternative enrollment endpoints MAY be supported at the registrar. The pledge will recognize whether its preferred enrollment option is supported by the domain registrar by sending a request to its preferred enrollment endpoint and evaluating the HTTP response status code.

The following list of endpoints provides an illustrative example for a domain registrar supporting several options for EST as well as for CMP to be used in BRSKI-AE. The listing contains the supported endpoints to which the pledge may connect for bootstrapping. This includes the voucher handling as well as the enrollment endpoints. The CMP related enrollment endpoints are defined as well-known URIs in CMP Updates [I-D.ietf-lamps-cmp-updates] and the Lightweight CMP profile [I-D.ietf-lamps-lightweight-cmp-profile].

```
</brski/voucherrequest>,ct=voucher-cms+json
</brski/voucher_status>,ct=json
</brski/enrollstatus>,ct=json
</est/cacerts>;ct=pkcs7-mime
</est/fullcmc>;ct=pkcs7-mime
</est/csrattrs>;ct=pkcs7-mime
</cmp/initialization>;ct=pkixcmp
</cmp/pl0>;ct=pkixcmp
</cmp/getcacerts>;ct=pkixcmp
</cmp/getcertreqtemplate>;ct=pkixcmp
```

## 5. Examples for signature-wrapping using existing enrollment protocols

This section maps the requirements to support proof-of-possession and proof-of-identity to selected existing enrollment protocols.

### 5.1. Instantiation to EST (informative)

When using EST [RFC7030], the following aspects and constraints need to be considered and the given extra requirements need to be fulfilled, which adapt Section 5.9.3 of BRSKI [RFC8995]:

- \* proof-of-possession is provided typically by using the specified PKCS#10 structure in the request. Together with Full PKI requests, also CRMF can be used.
- \* proof-of-identity needs to be achieved by signing the certification request object using the Full PKI Request option (including the /fullcmc endpoint). This provides sufficient information for the RA to authenticate the pledge as the origin of the request and to make an authorization decision on the received certification request. Note: EST references CMC [RFC5272] for the definition of the Full PKI Request. For proof-of-identity, the signature of the SignedData of the Full PKI Request is performed using the IDevID secret of the pledge.

Note: In this case the binding to the underlying TLS connection is not necessary.

- \* When the RA is temporarily not available, as per Section 4.2.3 of [RFC7030], an HTTP status code 202 should be returned by the registrar, and the pledge will repeat the initial Full PKI Request

## 5.2. Instantiation to CMP (normative if CMP is chosen)

Note: Instead of referring to CMP as specified in [RFC4210] and [I-D.ietf-lamps-cmp-updates], this document refers to the Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile] because the subset of CMP defined there is sufficient for the functionality needed here.

When using CMP, the following requirements SHALL be fulfilled:

- \* For proof-of-possession, the approach defined in Section 4.1.1 (based on CRMF) or Section 4.1.4 (based on PKCS#10) of the Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile] SHALL be applied.
- \* proof-of-identity SHALL be provided by using signature-based protection of the certification request message as outlined in Section 3.2. of [I-D.ietf-lamps-lightweight-cmp-profile] using the IDevID secret.
- \* When the Cert Response from the RA/CA is not available and if polling is supported, the registrar SHALL a Cert Waiting Response as specified in Sections 4.4 and 5.1.2 of [I-D.ietf-lamps-lightweight-cmp-profile].
- \* As far as requesting CA certificates or certificate request attributes is supported, they SHALL be implemented as specified in Sections 4.3.1 and 4.3.3 of [I-D.ietf-lamps-lightweight-cmp-profile].

TBD RFC Editor: please delete /\* ToDo: The following aspects need to be further specified: \* Whether to use /getcacerts or the caPubs and extraCerts fields to return trust anchor and CA Certificates \* Whether to use /getcertreqtemplate or modify the CRMF and use raVerified \* Whether to specify the usage of /p10 \*/

## 6. IANA Considerations

This document does not require IANA actions.

## 7. Security Considerations

The security considerations as laid out in BRSKI [RFC8995] apply for the discovery and voucher exchange as well as for the status exchange information.

The security considerations as laid out in the Lightweight CMP Profile [I-D.ietf-lamps-lightweight-cmp-profile] apply as far as CMP is used.

## 8. Acknowledgments

We would like to thank Brian E. Carpenter, Michael Richardson, and Giorgio Romanenghi for their input and discussion on use cases and call flows.

## 9. References

### 9.1. Normative References

[I-D.ietf-lamps-cmp-updates]

Brockhaus, H., Oheimb, D. V., and J. Gray, "Certificate Management Protocol (CMP) Updates", Work in Progress, Internet-Draft, draft-ietf-lamps-cmp-updates-17, 12 January 2022, <<https://www.ietf.org/archive/id/draft-ietf-lamps-cmp-updates-17.txt>>.

[I-D.ietf-lamps-lightweight-cmp-profile]

Brockhaus, H., Oheimb, D. V., and S. Fries, "Lightweight Certificate Management Protocol (CMP) Profile", Work in Progress, Internet-Draft, draft-ietf-lamps-lightweight-cmp-profile-10, 1 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-lamps-lightweight-cmp-profile-10.txt>>.

[IEEE.802.1AR\_2009]

IEEE, "IEEE Standard for Local and metropolitan area networks - Secure Device Identity", IEEE 802.1AR-2009, DOI 10.1109/ieeestd.2009.5367679, 28 December 2009, <<http://ieeexplore.ieee.org/servlet/opac?punumber=5367676>>.

[RFC2119]

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

- [RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <<https://www.rfc-editor.org/info/rfc4210>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <<https://www.rfc-editor.org/info/rfc8366>>.
- [RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <<https://www.rfc-editor.org/info/rfc8995>>.

## 9.2. Informative References

- [IEC-62351-9] International Electrotechnical Commission, "IEC 62351 - Power systems management and associated information exchange - Data and communications security - Part 9: Cyber security key management for power system equipment", IEC 62351-9, May 2017.
- [ISO-IEC-15118-2] International Standardization Organization / International Electrotechnical Commission, "ISO/IEC 15118-2 Road vehicles - Vehicle-to-Grid Communication Interface - Part 2: Network and application protocol requirements", ISO/IEC 15118-2, April 2014.
- [NERC-CIP-005-5] North American Reliability Council, "Cyber Security - Electronic Security Perimeter", CIP 005-5, December 2013.
- [Ocpp] Open Charge Alliance, "Open Charge Point Protocol 2.0.1 (Draft)", December 2019.
- [RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", RFC 2986, DOI 10.17487/RFC2986, November 2000, <<https://www.rfc-editor.org/info/rfc2986>>.

- [RFC4211] Schaad, J., "Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)", RFC 4211, DOI 10.17487/RFC4211, September 2005, <<https://www.rfc-editor.org/info/rfc4211>>.
- [RFC5272] Schaad, J. and M. Myers, "Certificate Management over CMS (CMC)", RFC 5272, DOI 10.17487/RFC5272, June 2008, <<https://www.rfc-editor.org/info/rfc5272>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [RFC5929] Altman, J., Williams, N., and L. Zhu, "Channel Bindings for TLS", RFC 5929, DOI 10.17487/RFC5929, July 2010, <<https://www.rfc-editor.org/info/rfc5929>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<https://www.rfc-editor.org/info/rfc7030>>.
- [RFC8894] Gutmann, P., "Simple Certificate Enrolment Protocol", RFC 8894, DOI 10.17487/RFC8894, September 2020, <<https://www.rfc-editor.org/info/rfc8894>>.
- [UNISIG-Subset-137]  
UNISIG, "Subset-137; ERTMS/ETCS On-line Key Management FFFIS; V1.0.0", December 2015, <[https://www.era.europa.eu/sites/default/files/filesystem/ertms/ccs\\_tsi\\_annex\\_a\\_-\\_mandatory\\_specifications/set\\_of\\_specifications\\_3\\_etcs\\_b3\\_r2\\_gsm-r\\_b1/index083\\_-\\_subset-137\\_v100.pdf](https://www.era.europa.eu/sites/default/files/filesystem/ertms/ccs_tsi_annex_a_-_mandatory_specifications/set_of_specifications_3_etcs_b3_r2_gsm-r_b1/index083_-_subset-137_v100.pdf)>.  
<http://www.kmc-subset137.eu/index.php/download/>

#### Appendix A. Using EST for certificate enrollment

When using EST with BRSKI, pledges interact via TLS with the domain registrar, which acts both as EST server and as registration authority (RA). The TLS connection is mutually authenticated, where the pledge uses its IDevID certificate issued by its manufacturer.

In order to provide a strong proof-of-origin of the certification request, EST has the option to include in the certification request the so-called `tls-unique` value [RFC5929] of the underlying TLS channel. This binding of the proof-of-identity of the TLS client, which is supposed to be the certificate requester, to the proof-of-possession for the private key is conceptually non-trivial and requires specific support by TLS implementations.

The registrar terminates the security association with the pledge at TLS level and thus the binding between the certification request and the authentication of the pledge. The EST server uses the authenticated pledge identity provided by the LDevID for checking the authorization of the pledge for the given certification request before issuing to the pledge a domain-specific certificate (LDevID certificate). This approach typically requires online or on-site availability of the RA for performing the final authorization decision for the certification request.

Using EST for BRSKI has the advantage that the mutually authenticated TLS connection established between the pledge and the registrar can be reused for protecting the message exchange needed for enrolling the LDevID certificate. This strongly simplifies the implementation of the enrollment message exchange.

Yet the use of TLS has the limitation that this cannot provide auditability nor end-to-end security for the certificate enrollment request because the TLS session is transient and terminates at the registrar. This is a problem in particular if the enrollment is done via multiple hops, part of which may not even be network-based.

A further limitation of using EST as the certificate enrollment protocol is that due to using PKCS#10 structures in enrollment requests, the only possible proof-of-possession method is a self-signature, which excludes requesting certificates for key types that do not support signing.

## Appendix B. Application examples

This informative annex provides some detail to the application examples listed in Section 1.3.

### B.1. Rolling stock

Rolling stock or railroad cars contain a variety of sensors, actuators, and controllers, which communicate within the railroad car but also exchange information between railroad cars building a train, with track-side equipment, and/or possibly with backend systems. These devices are typically unaware of backend system connectivity. Managing certificates may be done during maintenance cycles of the railroad car, but can already be prepared during operation. Preparation will include generating certification requests, which are collected and later forwarded for processing, once the railroad car is connected to the operator backend. The authorization of the certification request is then done based on the operator's asset/inventory information in the backend.

UNISIG has included a CMP profile for enrollment of TLS certificates of on-board and track-side components in the Subset-137 specifying the ETRAM/ETCS on-line key management for train control systems [UNISIG-Subset-137].

### B.2. Building automation

In building automation scenarios, a detached building or the basement of a building may be equipped with sensors, actuators, and controllers that are connected with each other in a local network but with only limited or no connectivity to a central building management system. This problem may occur during installation time but also during operation. In such a situation a service technician collects the necessary data and transfers it between the local network and the central building management system, e.g., using a laptop or a mobile phone. This data may comprise parameters and settings required in the operational phase of the sensors/actuators, like a component certificate issued by the operator to authenticate against other components and services.

The collected data may be provided by a domain registrar already existing in the local network. In this case connectivity to the backend PKI may be facilitated by the service technician's laptop. Alternatively, the data can also be collected from the pledges directly and provided to a domain registrar deployed in a different network as preparation for the operational phase. In this case, connectivity to the domain registrar may also be facilitated by the service technician's laptop.

### B.3. Substation automation

In electrical substation automation scenarios, a control center typically hosts PKI services to issue certificates for Intelligent Electronic Devices (IEDs) operated in a substation. Communication between the substation and control center is performed through a proxy/gateway/DMZ, which terminates protocol flows. Note that [NERC-CIP-005-5] requires inspection of protocols at the boundary of a security perimeter (the substation in this case). In addition, security management in substation automation assumes central support of several enrollment protocols in order to support the various capabilities of IEDs from different vendors. The IEC standard IEC62351-9 [IEC-62351-9] specifies mandatory support of two enrollment protocols: SCEP [RFC8894] and EST [RFC7030] for the infrastructure side, while the IED must only support one of the two.

### B.4. Electric vehicle charging infrastructure

For electric vehicle charging infrastructure, protocols have been defined for the interaction between the electric vehicle and the charging point (e.g., ISO 15118-2 [ISO-IEC-15118-2]) as well as between the charging point and the charging point operator (e.g. OCPP [OCPP]). Depending on the authentication model, unilateral or mutual authentication is required. In both cases the charging point uses an X.509 certificate to authenticate itself in TLS connections between the electric vehicle and the charging point. The management of this certificate depends, among others, on the selected backend connectivity protocol. In the case of OCPP, this protocol is meant to be the only communication protocol between the charging point and the backend, carrying all information to control the charging operations and maintain the charging point itself. This means that the certificate management needs to be handled in-band of OCPP. This requires the ability to encapsulate the certificate management messages in a transport-independent way. Authenticated self-containment will support this by allowing the transport without a separate enrollment protocol, binding the messages to the identity of the communicating endpoints.

### B.5. Infrastructure isolation policy

This refers to any case in which network infrastructure is normally isolated from the Internet as a matter of policy, most likely for security reasons. In such a case, limited access to external PKI services will be allowed in carefully controlled short periods of time, for example when a batch of new devices is deployed, and forbidden or prevented at other times.

#### B.6. Sites with insufficient level of operational security

The registration authority performing (at least part of) the authorization of a certification request is a critical PKI component and therefore requires higher operational security than components utilizing the issued certificates for their security features. CAs may also demand higher security in the registration procedures. Especially the CA/Browser forum currently increases the security requirements in the certificate issuance procedures for publicly trusted certificates. In case the on-site components of the target domain cannot be operated securely enough for the needs of a registration authority, this service should be transferred to an off-site backend component that has a sufficient level of security.

#### Appendix C. History of changes TBD RFC Editor: please delete

From IETF draft 04 -> IETF draft 05:

- \* David von Oheimb became the editor.
- \* Streamline wording, consolidate terminology, improve grammar, etc.
- \* Shift the emphasis towards supporting alternative enrollment protocols.
- \* Update the title accordingly - preliminary change to be approved.
- \* Move comments on EST and detailed application examples to informative annex.
- \* Move the remaining text of section 3 as two new sub-sections of section 1.

From IETF draft 03 -> IETF draft 04:

- \* Moved UC2 related parts defining the pledge in responder mode to a separate document. This required changes and adaptations in several sections. Main changes concerned the removal of the subsection for UC2 as well as the removal of the YANG model related text as it is not applicable in UC1.
- \* Updated references to the Lightweight CMP Profile.
- \* Added David von Oheimb as co-author.

From IETF draft 02 -> IETF draft 03:

- \* Housekeeping, deleted open issue regarding YANG voucher-request in UC2 as voucher-request was enhanced with additional leaf.
- \* Included open issues in YANG model in UC2 regarding assertion value agent-proximity and CSR encapsulation using SZTP sub module).

From IETF draft 01 -> IETF draft 02:

- \* Defined call flow and objects for interactions in UC2. Object format based on draft for JOSE signed voucher artifacts and aligned the remaining objects with this approach in UC2 .
- \* Terminology change: issue #2 pledge-agent -> registrar-agent to better underline agent relation.
- \* Terminology change: issue #3 PULL/PUSH -> pledge-initiator-mode and pledge-responder-mode to better address the pledge operation.
- \* Communication approach between pledge and registrar-agent changed by removing TLS-PSK (former section TLS establishment) and associated references to other drafts in favor of relying on higher layer exchange of signed data objects. These data objects are included also in the pledge-voucher-request and lead to an extension of the YANG module for the voucher-request (issue #12).
- \* Details on trust relationship between registrar-agent and registrar (issue #4, #5, #9) included in UC2.
- \* Recommendation regarding short-lived certificates for registrar-agent authentication towards registrar (issue #7) in the security considerations.
- \* Introduction of reference to agent signing certificate using SKID in agent signed data (issue #11).
- \* Enhanced objects in exchanges between pledge and registrar-agent to allow the registrar to verify agent-proximity to the pledge (issue #1) in UC2.
- \* Details on trust relationship between registrar-agent and pledge (issue #5) included in UC2.
- \* Split of use case 2 call flow into sub sections in UC2.

From IETF draft 00 -> IETF draft 01:

- \* Update of scope in Section 1.2 to include in which the pledge acts as a server. This is one main motivation for use case 2.
- \* Rework of use case 2 to consider the transport between the pledge and the pledge-agent. Addressed is the TLS channel establishment between the pledge-agent and the pledge as well as the endpoint definition on the pledge.
- \* First description of exchanged object types (needs more work)
- \* Clarification in discovery options for enrollment endpoints at the domain registrar based on well-known endpoints in Section 4.3 do not result in additional /.well-known URIs. Update of the illustrative example. Note that the change to /brski for the voucher related endpoints has been taken over in the BRSKI main document.
- \* Updated references.
- \* Included Thomas Werner as additional author for the document.

From individual version 03 -> IETF draft 00:

- \* Inclusion of discovery options of enrollment endpoints at the domain registrar based on well-known endpoints in Section 4.3 as replacement of section 5.1.3 in the individual draft. This is intended to support both use cases in the document. An illustrative example is provided.
- \* Missing details provided for the description and call flow in pledge-agent use case UC2, e.g. to accommodate distribution of CA certificates.
- \* Updated CMP example in Section 5 to use Lightweight CMP instead of CMP, as the draft already provides the necessary /.well-known endpoints.
- \* Requirements discussion moved to separate section in Section 3. Shortened description of proof of identity binding and mapping to existing protocols.
- \* Removal of copied call flows for voucher exchange and registrar discovery flow from [RFC8995] in Section 4 to avoid doubling or text or inconsistencies.

- \* Reworked abstract and introduction to be more crisp regarding the targeted solution. Several structural changes in the document to have a better distinction between requirements, use case description, and solution description as separate sections. History moved to appendix.

From individual version 02 -> 03:

- \* Update of terminology from self-contained to authenticated self-contained object to be consistent in the wording and to underline the protection of the object with an existing credential. Note that the naming of this object may be discussed. An alternative name may be attestation object.
- \* Simplification of the architecture approach for the initial use case having an offsite PKI.
- \* Introduction of a new use case utilizing authenticated self-contained objects to onboard a pledge using a commissioning tool containing a pledge-agent. This requires additional changes in the BRSKI call flow sequence and led to changes in the introduction, the application example, and also in the related BRSKI-AE call flow.
- \* Update of provided examples of the addressing approach used in BRSKI to allow for support of multiple enrollment protocols in Section 4.2.5.

From individual version 01 -> 02:

- \* Update of introduction text to clearly relate to the usage of IDevID and LDevID.
- \* Definition of the addressing approach used in BRSKI to allow for support of multiple enrollment protocols in Section 4.2.5. This section also contains a first discussion of an optional discovery mechanism to address situations in which the registrar supports more than one enrollment approach. Discovery should avoid that the pledge performs a trial and error of enrollment protocols.
- \* Update of description of architecture elements and changes to BRSKI in Section 4.1.
- \* Enhanced consideration of existing enrollment protocols in the context of mapping the requirements to existing solutions in Section 3 and in Section 5.

From individual version 00 -> 01:

- \* Update of examples, specifically for building automation as well as two new application use cases in Appendix B.
- \* Deletion of asynchronous interaction with MASA to not complicate the use case. Note that the voucher exchange can already be handled in an asynchronous manner and is therefore not considered further. This resulted in removal of the alternative path the MASA in Figure 1 and the associated description in Section 4.1.
- \* Enhancement of description of architecture elements and changes to BRSKI in Section 4.1.
- \* Consideration of existing enrollment protocols in the context of mapping the requirements to existing solutions in Section 3.
- \* New section starting Section 5 with the mapping to existing enrollment protocols by collecting boundary conditions.

#### Authors' Addresses

David von Oheimb (editor)  
Siemens AG  
Otto-Hahn-Ring 6  
81739 Munich  
Germany  
Email: david.von.oheimb@siemens.com  
URI: <https://www.siemens.com/>

Steffen Fries  
Siemens AG  
Otto-Hahn-Ring 6  
81739 Munich  
Germany  
Email: steffen.fries@siemens.com  
URI: <https://www.siemens.com/>

Hendrik Brockhaus  
Siemens AG  
Otto-Hahn-Ring 6  
81739 Munich  
Germany  
Email: hendrik.brockhaus@siemens.com  
URI: <https://www.siemens.com/>

Eliot Lear  
Cisco Systems  
Richtistrasse 7  
CH-8304 Wallisellen  
Switzerland  
Phone: +41 44 878 9200  
Email: [lear@cisco.com](mailto:lear@cisco.com)

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 7 September 2022

O. Friel  
Cisco  
R. Shekh-Yusef  
Auth0  
M. Richardson  
Sandelman Software Works  
6 March 2022

BRSKI Cloud Registrar  
draft-ietf-anima-brski-cloud-03

Abstract

This document specifies the behaviour of a BRSKI Cloud Registrar, and how a pledge can interact with a BRSKI Cloud Registrar when bootstrapping.

RFCEd REMOVE: It is being actively worked on at <https://github.com/anima-wg/brski-cloud>

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 7 September 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components

extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	3
1.1. Terminology . . . . .	3
1.2. Target Use Cases . . . . .	3
1.2.1. Owner Registrar Discovery . . . . .	4
1.2.2. Bootstrapping with no Owner Registrar . . . . .	4
2. Architecture . . . . .	5
2.1. Interested Parties . . . . .	6
2.2. Network Connectivity . . . . .	6
2.3. Pledge Certificate Identity Considerations . . . . .	6
3. Protocol Operation . . . . .	7
3.1. Pledge Requests Voucher from Cloud Registrar . . . . .	7
3.1.1. Cloud Registrar Discovery . . . . .	7
3.1.2. Pledge - Cloud Registrar TLS Establishment Details . . . . .	7
3.1.3. Pledge Issues Voucher Request . . . . .	8
3.2. Cloud Registrar Handles Voucher Request . . . . .	8
3.2.1. Pledge Ownership Lookup . . . . .	8
3.2.2. Cloud Registrar Redirects to Owner Registrar . . . . .	9
3.2.3. Cloud Registrar Issues Voucher . . . . .	9
3.3. Pledge Handles Cloud Registrar Response . . . . .	9
3.3.1. Redirect Response . . . . .	9
3.3.2. Voucher Response . . . . .	10
4. Protocol Details . . . . .	10
4.1. Voucher Request Redirected to Local Domain Registrar . . . . .	10
4.2. Voucher Request Handled by Cloud Registrar . . . . .	12
5. YANG extension for Voucher based redirect . . . . .	14
5.1. YANG Tree . . . . .	14
5.2. YANG Voucher . . . . .	15
6. IANA Considerations . . . . .	17
6.1. The IETF XML Registry . . . . .	17
6.2. The YANG Module Names Registry . . . . .	17
7. Security Considerations . . . . .	18
7.1. Issues with Security of HTTP Redirect . . . . .	18
7.2. Security Updates for the Pledge . . . . .	19
7.3. Trust Anchors for Cloud Registrar . . . . .	19
7.4. Issues with Redirect via Voucher . . . . .	20
8. References . . . . .	20
8.1. Normative References . . . . .	20
8.2. Informative References . . . . .	21
Authors' Addresses . . . . .	22

## 1. Introduction

Bootstrapping Remote Secure Key Infrastructures [BRSKI] specifies automated bootstrapping of an Autonomic Control Plane. BRSKI Section 2.7 describes how a pledge "MAY contact a well known URI of a cloud registrar if a local registrar cannot be discovered or if the pledge's target use cases do not include a local registrar".

This document further specifies use of a BRSKI cloud registrar and clarifies operations that are not sufficiently specified in BRSKI.

### 1.1. Terminology

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

This document uses the terms Pledge, Registrar, MASA, and Voucher from [BRSKI] and [RFC8366].

- \* Local Domain: The domain where the pledge is physically located and bootstrapping from. This may be different to the pledge owner's domain.
- \* Owner Domain: The domain that the pledge needs to discover and bootstrap with.
- \* Cloud Registrar: The default Registrar that is deployed at a URI that is well known to the pledge.
- \* Owner Registrar: The Registrar that is operated by the Owner, or the Owner's delegate. There may not be an Owner Registrar in all deployment scenarios.
- \* Local Domain Registrar: The Registrar discovered on the Local Domain. There may not be a Local Domain Registrar in all deployment scenarios.

### 1.2. Target Use Cases

Two high level use cases are documented here. There are more details provided in sections Section 4.1 and Section 4.2. While both use cases aid with incremental deployment of BRSKI infrastructure, for many smaller sites (such as teleworkers) no further infrastructure are expected.

The pledge is not expected to know which of these two situations it is in. The pledge determines this based upon signals that it receives from the Cloud Registrar. The Cloud Registrar is expected to make the determination based upon the identity presented by the pledge.

While a Cloud Registrar will typically handle all the devices of a particular product line from a particular manufacturer there are no restrictions on how the Cloud Registrar is horizontally (many sites) or vertically (more equipment at one site) scaled. It is also entirely possible that all devices sold by through a particular VAR might be preloaded with a configuration that changes the Cloud Registrar URL to point to a VAR. Such an effort would require unboxing each device in a controlled environment, but the provisioning could occur using a regular BRSKI or SZTP [RFC8572] process.

#### 1.2.1. Owner Registrar Discovery

A pledge is bootstrapping from a remote location with no local domain registrar (specifically: with no local infrastructure to provide for automated discovery), and needs to discover its owner registrar. The cloud registrar is used by the pledge to discover the owner registrar. The cloud registrar redirects the pledge to the owner registrar, and the pledge completes bootstrap against the owner registrar.

A typical example is an enduser deploying a pledge in a home or small branch office, where the pledge belongs to the enduser's employer. There is no local domain registrar, and the pledge needs to discover and bootstrap with the employer's registrar which is deployed in headquarters. For example, an enduser is deploying an IP phone in a home office and the phone needs to register to an IP PBX deployed in their employer's office.

#### 1.2.2. Bootstrapping with no Owner Registrar

A pledge is bootstrapping where the owner organization does not yet have an owner registrar deployed. The cloud registrar issues a voucher, and the pledge completes trust bootstrap using the cloud registrar. The voucher issued by the cloud includes domain information for the owner's [EST] service the pledge should use for certificate enrollment.

In one use case, an organization has an EST service deployed, but does not have yet a BRSKI capable Registrar service deployed. The pledge is deployed in the organizations domain, but does not discover a local domain, or owner, registrar. The pledge uses the cloud registrar to bootstrap, and the cloud registrar provides a voucher that includes instructions on finding the organization's EST service.

## 2. Architecture

The high level architecture is illustrated in Figure 1.

The pledge connects to the cloud registrar during bootstrap.

The cloud registrar may redirect the pledge to an owner registrar in order to complete bootstrap against the owner registrar.

If the cloud registrar issues a voucher itself without redirecting the pledge to an owner registrar, the cloud registrar will inform the pledge what domain to use for accessing EST services in the voucher response.

Finally, when bootstrapping against an owner registrar, this registrar may interact with a backend CA to assist in issuing certificates to the pledge. The mechanisms and protocols by which the registrar interacts with the CA are transparent to the pledge and are out-of-scope of this document.

The architecture shows the cloud registrar and MASA as being logically separate entities. The two functions could of course be integrated into a single service.

TWO CHOICES: 1. Cloud Registrar redirects to Owner Registrar 2. Cloud Registrar returns VOUCHER pinning Owner Register.

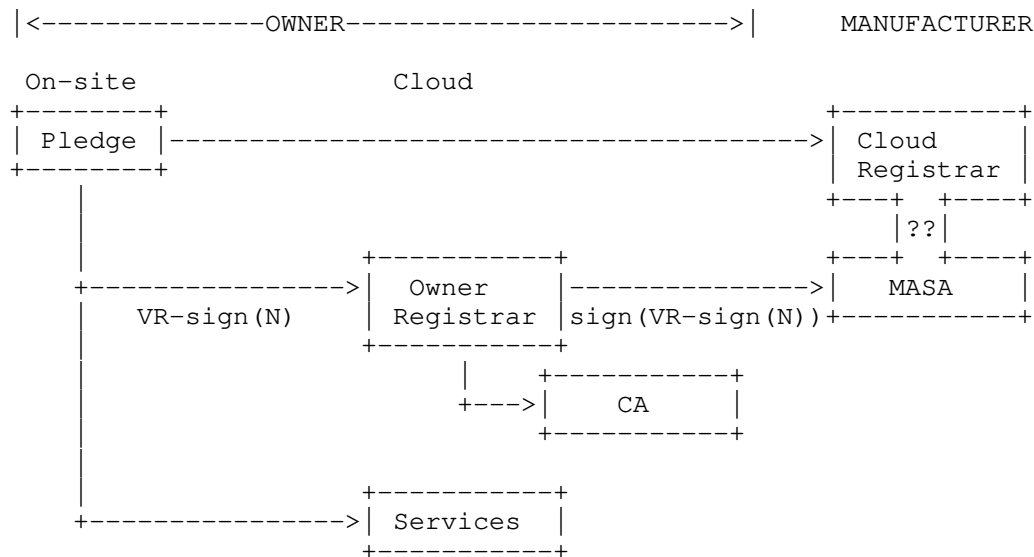


Figure 1: High Level Architecture

## 2.1. Interested Parties

1. OEM - Equipment manufacturer. Operate the MASA.
2. Network operator. Operate the Owner Registrar. Often operated by end owner (company), or by outsourced IT entity.
3. Network integrator. They operate a Cloud Registrar.

## 2.2. Network Connectivity

The assumption is that the pledge already has network connectivity prior to connecting to the cloud registrar. The pledge must have an IP address, must be able to make DNS queries, and must be able to send HTTP requests to the cloud registrar. The pledge operator has already connected the pledge to the network, and the mechanism by which this has happened is out of scope of this document.

## 2.3. Pledge Certificate Identity Considerations

BRSKI section 5.9.2 specifies that the pledge MUST send a CSR Attributes request to the registrar. The registrar MAY use this mechanism to instruct the pledge about the identities it should include in the CSR request it sends as part of enrollment. The registrar may use this mechanism to tell the pledge what Subject or Subject Alternative Name identity information to include in its CSR

request. This can be useful if the Subject must have a specific value in order to complete enrollment with the CA.

For example, the pledge may only be aware of its IDevID Subject which includes a manufacturer serial number, but must include a specific fully qualified domain name in the CSR in order to complete domain ownership proofs required by the CA.

As another example, the registrar may deem the manufacturer serial number in an IDevID as personally identifiable information, and may want to specify a new random opaque identifier that the pledge should use in its CSR.

### 3. Protocol Operation

#### 3.1. Pledge Requests Voucher from Cloud Registrar

##### 3.1.1. Cloud Registrar Discovery

BRSKI defines how a pledge MAY contact a well known URI of a cloud registrar if a local domain registrar cannot be discovered. Additionally, certain pledge types may never attempt to discover a local domain registrar and may automatically bootstrap against a cloud registrar.

The details of the URI are manufacturer specific, with BRSKI giving the example "brski-registrar.manufacturer.example.com".

The Pledge SHOULD be provided with the entire URL of the Cloud Registrar, including the path component, which is typically "/.well-known/brski/requestvoucher", but may be another value.

##### 3.1.2. Pledge - Cloud Registrar TLS Establishment Details

The pledge MUST use an Implicit Trust Anchor database (see [EST]) to authenticate the cloud registrar service. The Pledge can be done with pre-loaded trust-anchors that are used to validate the TLS connection. This can be using a public Web PKI trust anchors using [RFC6125] DNS-ID mechanisms, a pinned certification authority, or even a pinned raw public key. This is a local implementation decision.

The pledge MUST NOT establish a provisional TLS connection (see BRSKI section 5.1) with the cloud registrar.

The cloud registrar MUST validate the identity of the pledge by sending a TLS CertificateRequest message to the pledge during TLS session establishment. The cloud registrar MAY include a

certificate\_authorities field in the message to specify the set of allowed IDevID issuing CAs that pledges may use when establishing connections with the cloud registrar.

The cloud registrar MAY only allow connections from pledges that have an IDevID that is signed by one of a specific set of CAs, e.g. IDevIDs issued by certain manufacturers.

The cloud registrar MAY allow pledges to connect using self-signed identity certificates or using Raw Public Key [RFC7250] certificates.

### 3.1.3. Pledge Issues Voucher Request

After the pledge has established a full TLS connection with the cloud registrar and has verified the cloud registrar PKI identity, the pledge generates a voucher request message as outlined in BRSKI section 5.2, and sends the voucher request message to the cloud registrar.

### 3.2. Cloud Registrar Handles Voucher Request

The cloud registrar must determine pledge ownership. Once ownership is determined, or if no owner can be determined, then the registrar may:

- \* return a suitable 4xx or 5xx error response to the pledge if the registrar is unwilling or unable to handle the voucher request
- \* redirect the pledge to an owner register via 307 response code
- \* issue a voucher and return a 200 response code

#### 3.2.1. Pledge Ownership Lookup

The cloud registrar needs some suitable mechanism for knowing the correct owner of a connecting pledge based on the presented identity certificate. For example, if the pledge establishes TLS using an IDevID that is signed by a known manufacturing CA, the registrar could extract the serial number from the IDevID and use this to lookup a database of pledge IDevID serial numbers to owners.

Alternatively, if the cloud registrar allows pledges to connect using self-signed certificates, the registrar could use the thumbprint of the self-signed certificate to lookup a database of pledge self-signed certificate thumbprints to owners.

The mechanism by which the cloud registrar determines pledge ownership is out-of-scope of this document.

### 3.2.2. Cloud Registrar Redirects to Owner Registrar

Once the cloud registrar has determined pledge ownership, the cloud registrar may redirect the pledge to the owner registrar in order to complete bootstrap. Ownership registration will require the owner to register their local domain. The mechanism by which pledge owners register their domain with the cloud registrar is out-of-scope of this document.

The cloud registrar replies to the voucher request with a suitable HTTP 307 response code, including the owner's local domain in the HTTP Location header.

### 3.2.3. Cloud Registrar Issues Voucher

If the cloud registrar issues a voucher, it returns the voucher in a HTTP response with a 200 response code.

The cloud registrar MAY issue a 202 response code if it is willing to issue a voucher, but will take some time to prepare the voucher.

The voucher MUST include the "est-domain" field as defined below. This tells the pledge where the domain of the EST service to use for completing certificate enrollment.

The voucher MAY include the "additional-configuration" field.. This points the pledge to a URI where application specific additional configuration information may be retrieved. Pledge and Registrar behavior for handling and specifying the "additional-configuration" field is out-of-scope of this document.

## 3.3. Pledge Handles Cloud Registrar Response

### 3.3.1. Redirect Response

The cloud registrar returned a 307 response to the voucher request.

The pledge should restart the process using a new voucher request using the location provided in the HTTP redirect. Note if the pledge is able to validate the new server using a trust anchor found in its Implicit Trust Anchor database, then it MAY accept another 307 redirect. The pledge MUST never visit a location that it has already been to. If that happens then the pledge MUST fail the onboarding attempt and go back to the beginning, which includes listening to other sources of onboarding information as specified in [BRSKI] section 4.1 and 5.0.

The pledge should establish a provisional TLS connection with specified local domain registrar. The pledge should not use its Implicit Trust Anchor database for validating the local domain registrar identity. The pledge should send a voucher request message via the local domain registrar. When the pledge downloads a voucher, it can validate the TLS connection to the local domain registrar and continue with enrollment and bootstrap as per standard BRSKI operation.

### 3.3.2. Voucher Response

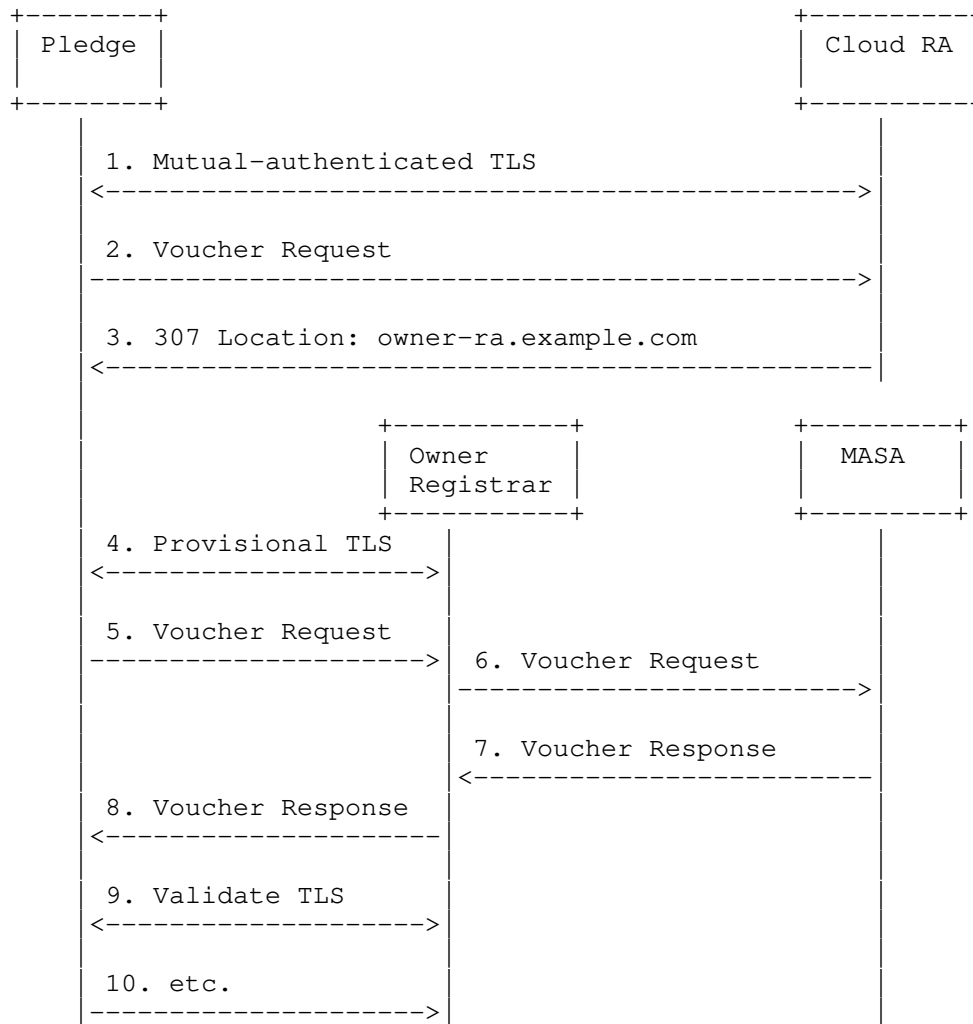
The cloud registrar returned a voucher to the pledge. The pledge should perform voucher verification as per standard BRSKI operation. The pledge should verify the voucher signature using the manufacturer-installed trust anchor(s), should verify the serial number in the voucher, and must verify any nonce information in the voucher.

The pledge should extract the "est-domain" field from the voucher, and should continue with EST enrollment as per standard BRSKI operation.

## 4. Protocol Details

### 4.1. Voucher Request Redirected to Local Domain Registrar

This flow illustrates the Owner Registrar Discovery flow. A pledge is bootstrapping in a remote location with no local domain registrar. The assumption is that the owner registrar domain is accessible and the pledge can establish a network connection with the owner registrar. This may require that the owner network firewall exposes the registrar on the public internet.



The process starts, in step 1, when the Pledge establishes a Mutual TLS channel with the Cloud RA using artifacts created during the manufacturing process of the Pledge.

In step 2, the Pledge sends a voucher request to the Cloud RA.

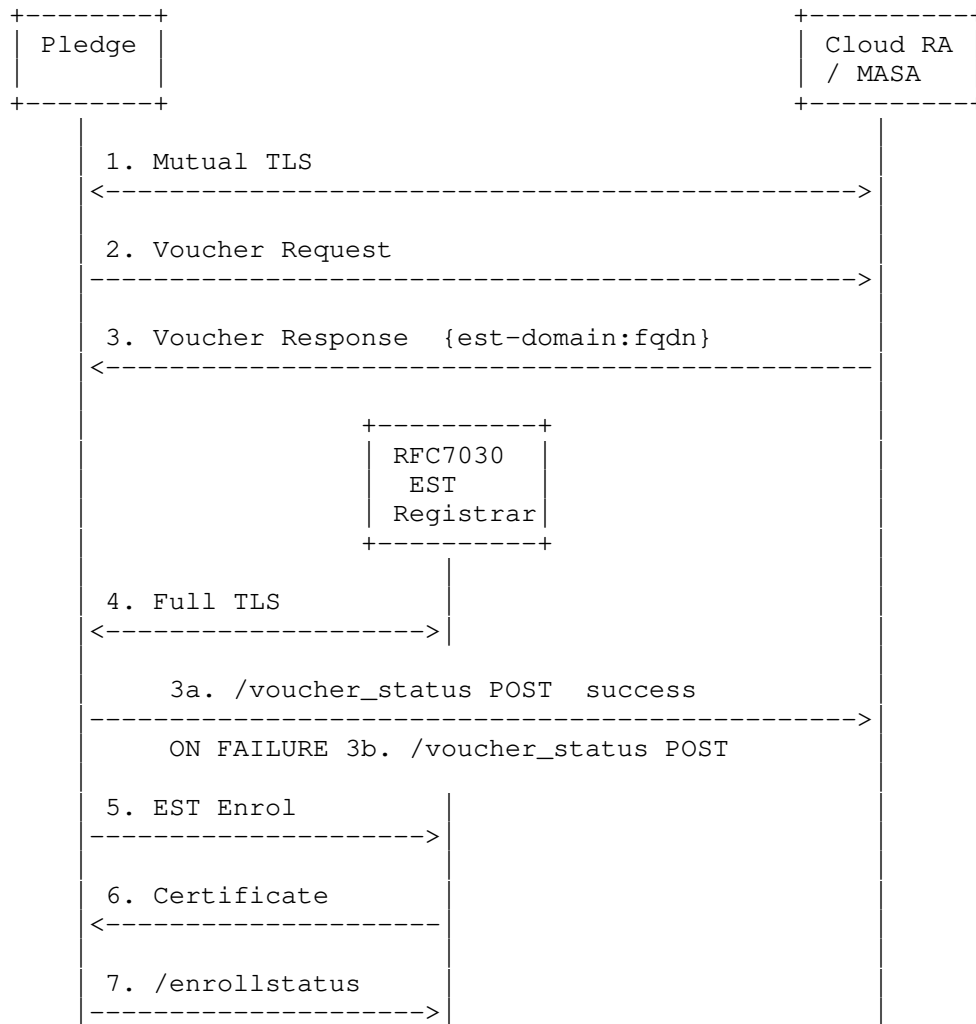
The Cloud RA completes pledge ownership lookup as outlined in Section 3.2.1, and determines the owner registrar domain. In step 3, the Cloud RA redirects the pledge to the owner registrar domain.

Steps 4 and onwards follow the standard BRSKI flow. The pledge establishes a provisional TLS connection with the owner registrar, and sends a voucher request to the owner registrar. The registrar forwards the voucher request to the MASA. Assuming the MASA issues a voucher, then the pledge validates the TLS connection with the registrar using the pinned-domain-cert from the voucher and completes the BRSKI flow.

#### 4.2. Voucher Request Handled by Cloud Registrar

The Voucher includes the EST domain to use for EST enroll. It is assumed services are accessed at that domain too. As trust is already established via the Voucher, the pledge does a full TLS handshake against the local RA indicated by the voucher response.

The returned voucher contains an attribute, "est-domain", defined in Section 5 below. The pledge is directed to continue enrollment using the EST registrar found at that URI. The pledge uses the pinned-domain-cert from the voucher to authenticate the EST registrar.



The process starts, in step 1, when the Pledge establishes a Mutual TLS channel with the Cloud RA/MASA using artifacts created during the manufacturing process of the Pledge. In step 2, the Pledge sends a voucher request to the Cloud RA/MASA, and in response the Pledge receives an [RFC8366] format voucher from the Cloud RA/MASA that includes its assigned EST domain in the est-domain attribute.

At this stage, the Pledge should be able to establish a TLS channel with the EST Registrar. The connection may involve crossing the Internet requiring a DNS lookup on the provided name. It may also be a local address that includes an IP address literal including both [RFC1918] and IPv6 Unique Local Address. The EST Registrar is

validated using the pinned-domain-cert value provided in the voucher as described in [BRSKI] section 5.6.2. This involves treating the artifact provided in the pinned-domain-cert as a trust anchor, and attempting to validate the EST Registrar from this anchor only.

There is a case where the pinned-domain-cert is the identical End-Entity (EE) Certificate as the EST Registrar. It also explicitly includes the case where the EST Registrar has a self-signed EE Certificate, but it may also be an EE certificate that is part of a larger PKI. If the certificate is not a self-signed or EE certificate, then the Pledge SHOULD apply [RFC6125] DNS-ID validation on the certificate against the URL provided in the est-domain attribute. If the est-domain was provided by with an IP address literal, then it is unlikely that it can be validated, and in that case, it is expected that either a self-signed certificate or an EE certificate will be pinned.

The Pledge also has the details it needs to be able to create the CSR request to send to the RA based on the details provided in the voucher.

In step 4, the Pledge establishes a TLS channel with the Cloud RA/MASA, and optionally the pledge should send a request, steps 3.a and 3.b, to the Cloud RA/MASA to inform it that the Pledge was able to establish a secure TLS channel with the EST Registrar.

The Pledge then follows that, in step 5, with an EST Enroll request with the CSR and obtains the requested certificate. The Pledge must validate that the issued certificate has the expected identifier obtained from the Cloud RA/MASA in step 3.

## 5. YANG extension for Voucher based redirect

An extension to the [RFC8366] voucher is needed for the case where the client will be redirected to a local EST Registrar.

### 5.1. YANG Tree

```
module: ietf-voucher-redirected
```

```
  grouping voucher-redirected-grouping
```

```
    +-- voucher
```

```
      +-- created-on                               yang:date-and-time
```

```
      +-- expires-on?                             yang:date-and-time
```

```
      +-- assertion                               enumeration
```

```
      +-- serial-number                           string
```

```
      +-- idevid-issuer?                          binary
```

```
      +-- pinned-domain-cert                     binary
```

```
      +-- domain-cert-revocation-checks?         boolean
```

```
      +-- nonce?                                 binary
```

```
      +-- last-renewal-date?                     yang:date-and-time
```

```
      +-- est-domain?                            ietf:uri
```

```
      +-- additional-configuration?              ietf:uri
```

## 5.2. YANG Voucher

```
<CODE BEGINS> file "ietf-voucher-redirected@2020-09-23.yang"
```

```
module ietf-voucher-redirected {
```

```
  yang-version 1.1;
```

```
  namespace
```

```
    "urn:ietf:params:xml:ns:yang:ietf-voucher-redirected";
```

```
  prefix "redirected";
```

```
  import ietf-restconf {
```

```
    prefix rc;
```

```
    description
```

```
      "This import statement is only present to access  
      the yang-data extension defined in RFC 8040.";
```

```
    reference "RFC 8040: RESTCONF Protocol";
```

```
  }
```

```
  import ietf-inet-types {
```

```
    prefix ietf;
```

```
    reference "RFC 6991: Common YANG Data Types";
```

```
  }
```

```
  import ietf-voucher {
```

```
    prefix "v";
```

```
  }
```

```
  organization
```

```
    "IETF ANIMA Working Group";
```

```
  contact
```

```
    "WG Web:  <http://tools.ietf.org/wg/anima/>
```

WG List: <mailto:anima@ietf.org>  
Author: Michael Richardson  
<mailto:mcr+ietf@sandelman.ca>  
Author: Owen Friel  
<mailto:ofriel@cisco.com>  
Author: Rifaat Shekh-Yusef  
<mailto:rifaat.ietf@gmail.com>;

description

"This module extends the base RFC8366 voucher format to include a redirect to an EST server to which enrollment should continue.

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in the module text are to be interpreted as described in BCP14, RFC 2119, and RFC8174.";

```
revision "2020-09-23" {  
  description  
    "Initial version";  
  reference  
    "RFC XXXX: Voucher Profile for Cloud redirected Devices";  
}
```

```
rc:yang-data voucher-redirected-artifact {  
  // YANG data template for a voucher.  
  uses voucher-redirected-grouping;  
}
```

```
// Grouping defined for future usage  
grouping voucher-redirected-grouping {  
  description  
    "Grouping to allow reuse/extensions in future work.";  
  
  uses v:voucher-artifact-grouping {  
  
    augment "voucher" {  
      description "Base the constrained voucher  
                  upon the regular one";  
      leaf est-domain {  
        type ietf:uri;  
        description  
          "The est-domain is a URL to which the Pledge should  
          continue doing enrollment rather than with the  
          Cloud Registrar.  
          The pinned-domain-cert contains a trust-anchor  
          which is to be used to authenticate the server  
          found at this URI.
```

```

    };
}
leaf additional-configuration {
    type ietf:uri;
    description
        "The additional-configuration attribute contains a
        URL to which the Pledge can retrieve additional
        configuration information.
        The contents of this URL are vendor specific.
        This is intended to do things like configure
        a VoIP phone to point to the correct hosted
        PBX, for example.";
}
}
}
}
}
<CODE ENDS>

```

## 6. IANA Considerations

## 6.1. The IETF XML Registry

This document registers one URI in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

```
{: newline="true"}
URI:
: urn:ietf:params:xml:ns:yang:ietf-voucher-redirected
```

Registrant Contact:  
: The ANIMA WG of the IETF.

XML:  
: N/A, the requested URI is an XML namespace.

## 6.2. The YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format defined in [RFC6020], the the following registration is requested:

```
{: newline="true"}
name:
: ietf-voucher-redirected

namespace:
: urn:ietf:params:xml:ns:yang:ietf-voucher-redirected

prefix:
: vch

reference:
: THIS DOCUMENT
```

## 7. Security Considerations

The Cloud-Registrar described in this document inherits all of the issues that are described in [BRSKI]. This includes dependency upon continued operation of the manufacturer provided MASA, as well as potential complications where a manufacturer might interfere with resale of a device.

In addition to the dependency upon the MASA, the successful enrollment of a device using a Cloud Registrar depends upon the correct and continued operation of this new service. This internet accessible service may be operated by the manufacturer and/or by one or more value-added-resellers. All of the considerations for operation of the MASA also apply to operation of the Cloud Registrar.

### 7.1. Issues with Security of HTTP Redirect

If the Redirect to Registrar method is used, as described in Section 4.1, there may be a series of 307 redirects. An example of why this might occur is that the manufacturer only knows that it resold the device to a particular value added reseller (VAR), and there may be a chain of such VARs. It is important the pledge avoid being drawn into a loop of redirects. This could happen if a VAR does not think they are authoritative for a particular device. A "helpful" programmer might instead decide to redirect back to the manufacturer in an attempt to restart at the top: perhaps there is another process that updates the manufacturer's database and this process is underway. Instead, the VAR MUST return a 404 error if it can not process the device. This will force the device to stop, timeout, and then try all mechanisms again.

There is another case where a connection problem may occur: when the pledge is behind a captive portal or an intelligent home gateway that provides access control on all connections. Captive portals that do not follow the requirements of [RFC8952] section 1 may forcibly

redirect HTTPS connections. While this is a deprecated practice as it breaks TLS in a way that most users can not deal with, it is still common in many networks.

On the first connection, the incorrect connection will be discovered because the Pledge will be unable to validate the connection to its cloud registrar via DNS-ID. That is, the certificate returned from the captive portal will not match.

At this point a network operator who controls the captive portal, noticing the connection to what seems a legitimate destination (the cloud registrar), may then permit that connection. This enables the first connection to go through.

The connection is then redirected to the Registrar, either via 307, or via est-domain in a voucher. If it is a 307 redirect, then a provisional TLS connection will be initiated, and it will succeed. The provisional TLS connection does not do [RFC6125] DNS-ID validation at the beginning of the connection, so a forced redirection to a captive portal system will not be detected. The subsequent BRSKI POST of a voucher will most likely be met by a 404 or 500 HTTP code. As the connection is provisional, the pledge will be unable to determine this.

It is RECOMMENDED therefore that the pledge look for [RFC8910] attributes in DHCP, and if present, use the [RFC8908] API to learn if it is captive.

## 7.2. Security Updates for the Pledge

Unlike many other uses of BRSKI, in the Cloud Registrar case it is assumed that the Pledge has connected to a network on which there is addressing and connectivity, but there is no other local configuration available.

There is another advantage to being online: the pledge may be able to contact the manufacturer before onboarding in order to apply the latest firmware updates. This may also include updates to the Implicit list of Trust Anchors. In this way, a Pledge that may have been in a dusty box in a warehouse for a long time can be updated to the latest (exploit-free) firmware before attempting onboarding.

## 7.3. Trust Anchors for Cloud Registrar

The Implicit TA database is used to authenticate the Cloud Registrar. This list is built-in by the manufacturer along with a DNS name to which to connect. (The manufacturer could even build in IP addresses as a last resort)

The Cloud Registrar does not have have a certificate that can be validated using a public (WebPKI) anchor. The pledge may have any kind of Trust Anchor built in: from full multi-level WebPKI to the single self-signed certificate used by the Cloud Registrar. There are many tradeoffs to having more or less of the PKI present in the Pledge, which is addresses in part in [I-D.richardson-t2trg-idevid-considerations] in sections 3 and 5.

#### 7.4. Issues with Redirect via Voucher

The second redirect case is handled by returning a special extension in the voucher. The Cloud Registrar actually does all of the voucher processing as specified in [BRSKI]. In this case, the Cloud Registrar may be operated by the same entity as the MASA, and it might even be combined into a single server. Whether or not this is the case, it behaves as if it was separate.

It may be the case that one or more 307-Redirects have taken the Pledge from the built-in Cloud Registrar to one operated by a VAR.

When the Pledge is directed to the Owner's [EST] Registrar, the Pledge validates the TLS connection with this server using the "pinned-domain-cert" attribute in the voucher. There is no provisional TLS connection, and therefore there are no risks associated with being behind a captive portal.

## 8. References

### 8.1. Normative References

- [BRSKI] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <<https://www.rfc-editor.org/info/rfc8995>>.
- [EST] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<https://www.rfc-editor.org/info/rfc7030>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <<https://www.rfc-editor.org/info/rfc8366>>.

## 8.2. Informative References

- [I-D.richardson-t2trg-idevid-considerations]  
Richardson, M., "A Taxonomy of operational security considerations for manufacturer installed keys and Trust Anchors", Work in Progress, Internet-Draft, draft-richardson-t2trg-idevid-considerations-06, 3 February 2022, <<https://www.ietf.org/archive/id/draft-richardson-t2trg-idevid-considerations-06.txt>>.
- [IEEE802.1AR]  
IEEE Standard, ., "IEEE 802.1AR Secure Device Identifier", 2018, <<http://standards.ieee.org/findstds/standard/802.1AR-2018.html>>.
- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G. J., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <<https://www.rfc-editor.org/info/rfc1918>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <<https://www.rfc-editor.org/info/rfc7250>>.

- [RFC8572] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <<https://www.rfc-editor.org/info/rfc8572>>.
- [RFC8908] Pauly, T., Ed. and D. Thakore, Ed., "Captive Portal API", RFC 8908, DOI 10.17487/RFC8908, September 2020, <<https://www.rfc-editor.org/info/rfc8908>>.
- [RFC8910] Kumari, W. and E. Kline, "Captive-Portal Identification in DHCP and Router Advertisements (RAs)", RFC 8910, DOI 10.17487/RFC8910, September 2020, <<https://www.rfc-editor.org/info/rfc8910>>.
- [RFC8952] Larose, K., Dolson, D., and H. Liu, "Captive Portal Architecture", RFC 8952, DOI 10.17487/RFC8952, November 2020, <<https://www.rfc-editor.org/info/rfc8952>>.

## Authors' Addresses

Owen Friel  
Cisco  
Email: [ofriel@cisco.com](mailto:ofriel@cisco.com)

Rifaat Shekh-Yusef  
Auth0  
Email: [rifaat.s.ietf@gmail.com](mailto:rifaat.s.ietf@gmail.com)

Michael Richardson  
Sandelman Software Works  
Email: [mcr+ietf@sandelman.ca](mailto:mcr+ietf@sandelman.ca)

anima Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 8, 2021

M. Richardson  
Sandelman Software Works  
P. van der Stok  
vanderstok consultancy  
P. Kampanakis  
Cisco Systems  
February 04, 2021

Constrained Join Proxy for Bootstrapping Protocols  
draft-ietf-anima-constrained-join-proxy-02

Abstract

This document defines a protocol to securely assign a pledge to a domain, represented by a Registrar, using an intermediary node between pledge and Registrar. This intermediary node is known as a "constrained Join Proxy".

This document extends the work of [I-D.ietf-anima-bootstrapping-keyinfra] by replacing the Circuit-proxy by a stateless/stateful constrained (CoAP) Join Proxy. It transports join traffic from the pledge to the Registrar without requiring per-client state.

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 August 8, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	4
3. Requirements Language . . . . .	4
4. Join Proxy functionality . . . . .	4
5. Join Proxy specification . . . . .	5
5.1. Statefull Join Proxy . . . . .	5
5.2. Stateless Join Proxy . . . . .	6
5.3. Stateless Message structure . . . . .	8
6. Comparison of stateless and statefull modes . . . . .	9
7. Discovery . . . . .	10
7.1. Pledge discovery of Registrar . . . . .	11
7.1.1. CoAP discovery . . . . .	11
7.1.2. Autonomous Network . . . . .	11
7.1.3. 6tisch discovery . . . . .	11
7.2. Pledge discovers Join Proxy . . . . .	11
7.2.1. Autonomous Network . . . . .	12
7.2.2. CoAP discovery . . . . .	12
7.3. Join Proxy discovers Registrar join port . . . . .	12
7.3.1. CoAP discovery . . . . .	12
8. Security Considerations . . . . .	13
9. IANA Considerations . . . . .	13
9.1. Resource Type registry . . . . .	13
10. Acknowledgements . . . . .	14
11. Contributors . . . . .	14
12. Changelog . . . . .	14
12.1. 01 to 02 . . . . .	14
12.2. 00 to 01 . . . . .	14
12.3. 00 to 00 . . . . .	14
13. References . . . . .	14
13.1. Normative References . . . . .	14
13.2. Informative References . . . . .	16
Appendix A. Stateless Proxy payload examples . . . . .	17
Authors' Addresses . . . . .	17

## 1. Introduction

Enrolment of new nodes into networks with enrolled nodes present is described in [I-D.ietf-anima-bootstrapping-keyinfra] ("BRSKI") and makes use of Enrolment over Secure Transport (EST) [RFC7030] with [RFC8366] vouchers to securely enroll devices. BRSKI connects new devices ("pledges") to "Registrars" via a Join Proxy.

The specified solutions use https and may be too large in terms of code space or bandwidth required for constrained devices. Constrained devices possibly part of constrained networks [RFC7228] typically implement the IPv6 over Low-Power Wireless personal Area Networks (6LoWPAN) [RFC4944] and Constrained Application Protocol (CoAP) [RFC7252].

CoAP can be run with the Datagram Transport Layer Security (DTLS) [RFC6347] as a security protocol for authenticity and confidentiality of the messages. This is known as the "coaps" scheme. A constrained version of EST, using Coap and DTLS, is described in [I-D.ietf-ace-coap-est]. The {I-D.ietf-anima-constrained-voucher} describes the BRSKI extensions to the Registrar.

DTLS is a client-server protocol relying on the underlying IP layer to perform the routing between the DTLS Client and the DTLS Server. However, the new "joining" device will not be IP routable until it is authenticated to the network. A new "joining" device can only initially use a link-local IPv6 address to communicate with a neighbour node using neighbour discovery [RFC6775] until it receives the necessary network configuration parameters. However, before the device can receive these configuration parameters, it needs to authenticate itself to the network to which it connects. IPv6 routing is necessary to establish a connection between joining device and the Registrar.

A DTLS connection is required between Pledge and Registrar.

This document specifies a new form of Join Proxy and protocol to act as intermediary between joining device and Registrar to establish a connection between joining device and Registrar.

This document is very much inspired by text published earlier in [I-D.kumar-dice-dtls-relay]. [I-D.richardson-anima-state-for-joinrouter] outlined the various options for building a join proxy. [I-D.ietf-anima-bootstrapping-keyinfra] adopted only the Circuit Proxy method (1), leaving the other methods as future work. This document standardizes the CoAP/DTLS (method 4).

## 2. Terminology

The following terms are defined in [RFC8366], and are used identically as in that document: artifact, imprint, domain, Join Registrar/Coordinator (JRC), Manufacturer Authorized Signing Authority (MASA), pledge, Trust of First Use (TOFU), and Voucher.

## 3. Requirements Language

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

## 4. Join Proxy functionality

As depicted in the Figure 1, the joining Device, or pledge (P), in an LLN mesh can be more than one hop away from the Registrar (R) and not yet authenticated into the network.

In this situation, it can only communicate one-hop to its nearest neighbour, the Join Proxy (J) using their link-local IPv6 addresses. However, the Pledge (P) needs to communicate with end-to-end security with a Registrar hosting the Registrar (R) to authenticate and get the relevant system/network parameters. If the Pledge (P) initiates a DTLS connection to the Registrar whose IP address has been pre-configured, then the packets are dropped at the Join Proxy (J) since the Pledge (P) is not yet admitted to the network or there is no IP routability to Pledge (P) for any returned messages.

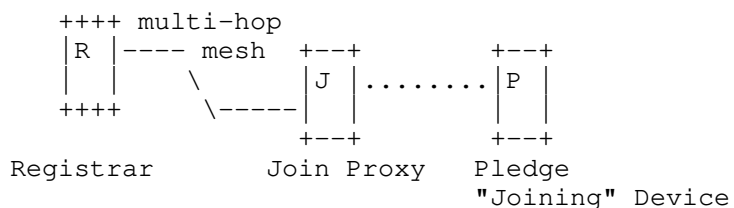


Figure 1: multi-hop enrolment.

Without routing the Pledge (P) cannot establish a secure connection to the Registrar (R) in the network assuming appropriate credentials are exchanged out-of-band, e.g. a hash of the Pledge (P)'s raw public key could be provided to the Registrar (R).

Furthermore, the Pledge (P) may be unaware of the IP address of the Registrar (R) to initiate a DTLS connection and perform authentication.

To overcome the problems with non-routability of DTLS packets and/or discovery of the destination address of the EST Server to contact, the Join Proxy is introduced. This Join Proxy functionality is configured into all authenticated devices in the network which may act as the Join Proxy for newly joining nodes. The Join Proxy allows for routing of the packets from the Pledge using IP routing to the intended Registrar.

## 5. Join Proxy specification

A Join Proxy can operate in two modes:

- o Statefull mode
- o Stateless mode

### 5.1. Statefull Join Proxy

In stateful mode, the joining node forwards the DTLS messages to the Registrar.

Assume that the Pledge does not know the IP address of the Registrar it needs to contact. The Join Proxy has been enrolled via the Registrar and consequently knows the IP address and port of the Registrar. The Pledge first discovers and selects the most appropriate Join Proxy. (Discovery can be based upon [I-D.ietf-anima-bootstrapping-keyinfra] section 4.3, or via DNS-SD service discovery [RFC6763]). The Pledge initiates its request as if the Join Proxy is the intended Registrar. The Join Proxy receives the message at a discoverable "Join" port. The Join Proxy changes the IP packet (without modifying the DTLS message) by modifying both the source and destination addresses to forward the message to the intended Registrar. The Join Proxy maintains a 4-tuple array to translate the DTLS messages received from the Registrar and forward it to the EST Client. This is a form of Network Address translation, where the Join Proxy acts as a forward proxy. In Figure 2 the various steps of the message flow are shown, with 5684 being the standard coaps port:

Pledge (P)	Join Proxy (J)	Registrar (R)	Message	
			Src_IP:port	Dst_IP:port
--ClientHello-->			IP_P:p_P	IP_Ja:p_J
			IP_Jb:p_Jb	IP_R:5684
		<--ServerHello-- :	IP_R:5684	IP_Jb:p_Jb
<--ServerHello-- :		:	IP_Ja:p_J	IP_P:p_P
		:	:	:
			:	:
--Finished-->		:	IP_P:p_P	IP_Ja:p_J
			IP_Jb:p_Jb	IP_R:5684
		<--Finished-- :	IP_R:5684	IP_Jb:p_Jb
			IP_Ja:p_J	IP_P:p_P
	:	:	:	:

IP\_P:p\_P = Link-local IP address and port of Pledge (DTLS Client)

IP\_R:5684 = Global IP address and coaps port of Registrar

IP\_Ja:P\_J = Link-local IP address and join port of Join Proxy

IP\_Jb:p\_Rb = Global IP address and client port of Join proxy

Figure 2: constrained statefull joining message flow with Registrar address known to Join Proxy.

## 5.2. Stateless Join Proxy

The stateless Join Proxy aims to minimize the requirements on the constrained Join Proxy device. Stateless operation requires no memory in the Join Proxy device, but may also reduce the CPU impact as the device does not need to search through a state table.

If an untrusted Pledge that can only use link-local addressing wants to contact a trusted Registrar, and the Registrar is more than one hop away, it sends the DTLS message to the Join Proxy.

When a Pledge attempts a DTLS connection to the Join Proxy, it uses its link-local IP address as its IP source address. This message is transmitted one-hop to a neighbouring (Join Proxy) node. Under normal circumstances, this message would be dropped at the neighbour node since the Pledge is not yet IP routable or is not yet authenticated to send messages through the network. However, if the neighbour device has the Join Proxy functionality enabled, it routes the DTLS message to its Registrar of choice.

The Join Proxy extends this message into a new type of message called Join ProxY (JPY) message and sends it on to the Registrar.

The JPY message payload consists of two parts:

- o Header (H) field: consisting of the source link-local address and port of the Pledge (P), and
- o Contents (C) field: containing the original DTLS message.

On receiving the JPY message, the Registrar retrieves the two parts.

The Registrar transiently stores the Header field information. The Registrar uses the Contents field to execute the Registrar functionality. However, when the Registrar replies, it also extends its DTLS message with the header field in a JPY message and sends it back to the Join Proxy. The Registrar SHOULD NOT assume that it can decode the Header Field, it should simply repeat it when responding. The Header contains the original source link-local address and port of the pledge from the transient state stored earlier and the Contents field contains the DTLS message.

On receiving the JPY message, the Join Proxy retrieves the two parts. It uses the Header field to route the DTLS message retrieved from the Contents field to the Pledge.

In this scenario, both the Registrar and the Join Proxy use discoverable "Join" ports.

The Figure 3 depicts the message flow diagram:

EST	Client (P)	Join Proxy (J)	Registrar (R)	Message	
				Src_IP:port	Dst_IP:port
		--ClientHello-->		IP_P:p_P	IP_Ja:p_Ja
		--JPY[H(IP_P:p_P),-->		IP_Jb:p_Jb	IP_R:p_Ra
		C(ClientHello)]			
		<--JPY[H(IP_P:p_P),--		IP_R:p_Ra	IP_Jb:p_Jb
		C(ServerHello)]			
		<--ServerHello--		IP_Ja:p_Ja	IP_P:p_P
		:		:	:
		:		:	:
		--Finished-->		IP_P:p_P	IP_Ja:p_Ja
		--JPY[H(IP_P:p_P),-->		IP_Jb:p_Jb	IP_R:p_Ra
		C(Finished)]			
		<--JPY[H(IP_P:p_P),--		IP_R:p_Ra	IP_Jb:p_Jb
		C(Finished)]			
		<--Finished--		IP_Ja:p_Ja	IP_P:p_P
		:		:	:

IP\_P:p\_P = Link-local IP address and port of the Pledge

IP\_R:p\_Ra = Global IP address and join port of Registrar

IP\_Ja:p\_Ja = Link-local IP address and join port of Join Proxy

IP\_Jb:p\_Jb = Global IP address and port of Join Proxy

JPY[H()],C()] = Join Proxy message with header H and content C

Figure 3: constrained stateless joining message flow.

### 5.3. Stateless Message structure

The JPY message is constructed as a payload with media-type application/cbor

Header and Contents fields together are one cbor array of 5 elements:

1. header field: containing a CBOR array [RFC7049] with the pledge IPv6 Link Local address as a cbor byte string, the pledge's UDP port number as a CBOR integer, the IP address family (IPv4/IPv6) as a cbor integer, and the proxy's ifindex or other identifier for the physical port as cbor integer. The header field is not DTLS encrypted.
2. Content field: containing the DTLS encrypted payload as a CBOR byte string.

The join\_proxy cannot decrypt the DTLS encrypted payload and has no knowledge of the transported media type.

```
JPY_message =  
[  
    ip      : bstr,  
    port    : int,  
    family   : int,  
    index    : int  
    payload : bstr  
]
```

Figure 4: CDDL representation of JPY message

The content fields are DTLS encrypted. In CBOR diagnostic notation the payload JPY[H(IP\_P:p\_P)], will look like:

```
[h'IP_p', p_P, family, ident, h'DTLS-content']
```

Examples are shown in Appendix A.

## 6. Comparison of stateless and statefull modes

The stateful and stateless mode of operation for the Join Proxy have their advantages and disadvantages. This section should enable to make a choice between the two modes based on the available device resources and network bandwidth.

Properties	Stateful mode	Stateless mode
State Information	The Join Proxy needs additional storage to maintain mapping between the address and port number of the pledge and those of the Registrar.	No information is maintained by the Join Proxy. Registrar needs to store the packet header.
Packet size	The size of the forwarded message is the same as the original message.	Size of the forwarded message is bigger than the original, it includes additional source and destination addresses.
Specification complexity	The Join Proxy needs additional functionality to maintain state information, and modify the source and destination addresses of the DTLS handshake messages	New JPY message to encapsulate DTLS message The Registrar and the Join Proxy have to understand the JPY message in order to process it.
Ports	Join Proxy needs discoverable "Join" port	Join Proxy and Registrar need discoverable "Join" ports

Figure 5: Comparison between stateful and stateless mode

## 7. Discovery

It is assumed that Join Proxy seamlessly provides a coaps connection between Pledge and coaps Registrar. In particular this section replaces section 4.2 of [I-D.ietf-anima-bootstrapping-keyinfra].

The discovery follows two steps:

1. The pledge is one hop away from the Registrar. The pledge discovers the link-local address of the Registrar as described in {I-D.ietf-ace-coap-est}. From then on, it follows the BRSKI process as described in {I-D.ietf-ace-coap-est}, using link-local addresses.
2. The pledge is more than one hop away from a relevant Registrar, and discovers the link-local address and join port of a Join

Proxy. The pledge then follows the BRSKI procedure using the link-local address of the Join Proxy.

### 3. The stateless Join Proxy discovers the join port of the Registrar

Once a pledge is enrolled, it may function as Join Proxy. The Join Proxy functions are advertised as described below. In principle, the Join Proxy functions are offered via a "join" port, and not the standard coaps port. Also the Registrar offer a "join" port to which the stateless join proxy sends the JPY message. The Join Proxy and Registrar MUST show the extra join port number when repending to the .well-known/core request addressed to the standard coap/coaps port.

Three discovery cases are discussed: coap discovery, 6tisch discovery and GRASP discovery.

#### 7.1. Pledge discovery of Registrar

The Pledge and Join Proxy are assumed to communicate via Link-Local addresses.

##### 7.1.1. CoAP discovery

The discovery of the coaps Registrar, using coap discovery, by the Join Proxy follows section 6 of [I-D.ietf-ace-coap-est]. The extension to discover the additional port needed by the stateless proxy is described in Section 7.2.2.

##### 7.1.2. Autonomous Network

In the context of autonomous networks, the Join Proxy uses the DULL GRASP M\_FLOOD mechanism to announce itself. Section 4.1.1 of [I-D.ietf-anima-bootstrapping-keyinfra] discusses this in more detail. The Registrar announces itself using ACP instance of GRASP using M\_FLOOD messages. Autonomous Network Join Proxies MUST support GRASP discovery of Registrar as decribed in section 4.3 of [I-D.ietf-anima-bootstrapping-keyinfra] .

##### 7.1.3. 6tisch discovery

The discovery of Registrar by the pledge uses the enhanced beacons as discussed in [I-D.ietf-6tisch-enrollment-enhanced-beacon].

#### 7.2. Pledge discovers Join Proxy

### 7.2.1. Autonomous Network

The pledge MUST listen for GRASP M\_FLOOD [I-D.ietf-anima-grasp] announcements of the objective: "AN\_Proxy". See section Section 4.1.1 [I-D.ietf-anima-bootstrapping-keyinfra] for the details of the objective.

### 7.2.2. CoAP discovery

In the context of a coap network without Autonomous Network support, discovery follows the standard coap policy. The Pledge can discover a Join Proxy by sending a link-local multicast message to ALL CoAP Nodes with address FF02::FD. Multiple or no nodes may respond. The handling of multiple responses and the absence of responses follow section 4 of [I-D.ietf-anima-bootstrapping-keyinfra].

The join port of the Join Proxy is discovered by sending a GET request to `"/.well-known/core"` including a resource type (rt) parameter with the value `"brski-proxy"` [RFC6690]. Upon success, the return payload will contain the join port.

The example below shows the discovery of the join port of the Join Proxy.

```
REQ: GET coap://[FF02::FD]/.well-known/core?rt=brski-proxy
```

```
RES: 2.05 Content
```

```
<coaps://[IP_address]:join-port>; rt="brski-proxy"
```

Port numbers are assumed to be the default numbers 5683 and 5684 for coap and coaps respectively (sections 12.6 and 12.7 of [RFC7252] when not shown in the response. Discoverable port numbers are usually returned for Join Proxy resources in the `<href>` of the payload (see section 5.1 of [I-D.ietf-ace-coap-est]).

## 7.3. Join Proxy discovers Registrar join port

### 7.3.1. CoAP discovery

The stateless Join Proxy can discover the join port of the Registrar by sending a GET request to `"/.well-known/core"` including a resource type (rt) parameter with the value `"join-proxy"` [RFC6690]. Upon success, the return payload will contain the join Port of the Registrar.

```
REQ: GET coap://[IP_address]/.well-known/core?rt=brski-proxy
```

```
RES: 2.05 Content
```

```
<coaps://[IP_address]:join-port>; rt="join-proxy"
```

The discoverable port numbers are usually returned for Join Proxy resources in the <href> of the payload (see section 5.1 of [I-D.ietf-ace-coap-est]).

## 8. Security Considerations

It should be noted here that the contents of the CBOR map used to convey return address information is not protected. However, the communication is between the Proxy and a known registrar are over the already secured portion of the network, so are not visible to eavesdropping systems.

All of the concerns in [I-D.ietf-anima-bootstrapping-keyinfra] section 4.1 apply. The pledge can be deceived by malicious AN\_Proxy announcements. The pledge will only join a network to which it receives a valid [RFC8366] voucher.

If the proxy/Registrar was not over a secure network, then an attacker could change the cbor array, causing the pledge to send traffic to another node. If the such scenario needed to be supported, then it would be reasonable for the Proxy to encrypt the CBOR array using a locally generated symmetric key. The Registrar would not be able to examine the result, but it does not need to do so. This is a topic for future work.

## 9. IANA Considerations

This document needs to create a registry for key indices in the CBOR map. It should be given a name, and the amending formula should be IETF Specification.

### 9.1. Resource Type registry

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

rt="brski-proxy". This BRSKI resource is used to query and return the supported BRSKI port of the Join Proxy.

rt="join-proxy". This BRSKI resource is used to query and return the supported BRSKI port of the Registrar.

## 10. Acknowledgements

Many thanks for the comments by Brian Carpenter and Esko Dijk.

## 11. Contributors

Sandeep Kumar, Sye loong Keoh, and Oscar Garcia-Morchon are the co-authors of the draft-kumar-dice-dtls-relay-02. Their draft has served as a basis for this document. Much text from their draft is copied over to this draft.

## 12. Changelog

### 12.1. 01 to 02

- o Discovery of Join Proxy and Registrar ports

### 12.2. 00 to 01

- o Registrar used throughout instead of EST server
- o Emphasized additional Join Proxy port for Join Proxy and Registrar
- o updated discovery accordingly
- o updated stateless Join Proxy JPY header
- o JPY header described with CDDL
- o Example simplified and corrected

### 12.3. 00 to 00

- o copied from vanderstok-anima-constrained-join-proxy-05

## 13. References

### 13.1. Normative References

[I-D.ietf-6tisch-enrollment-enhanced-beacon]  
Dujovne, D. and M. Richardson, "IEEE 802.15.4 Information Element encapsulation of 6TiSCH Join and Enrollment Information", draft-ietf-6tisch-enrollment-enhanced-beacon-14 (work in progress), February 2020.

- [I-D.ietf-ace-coap-est]  
Stok, P., Kampanakis, P., Richardson, M., and S. Raza,  
"EST over secure CoAP (EST-coaps)", draft-ietf-ace-coap-  
est-18 (work in progress), January 2020.
- [I-D.ietf-anima-bootstrapping-keyinfra]  
Pritikin, M., Richardson, M., Eckert, T., Behringer, M.,  
and K. Watsen, "Bootstrapping Remote Secure Key  
Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-  
keyinfra-45 (work in progress), November 2020.
- [I-D.ietf-anima-constrained-voucher]  
Richardson, M., Stok, P., and P. Kampanakis, "Constrained  
Voucher Artifacts for Bootstrapping Protocols", draft-  
ietf-anima-constrained-voucher-09 (work in progress),  
November 2020.
- [I-D.ietf-anima-grasp]  
Bormann, C., Carpenter, B., and B. Liu, "A Generic  
Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-  
grasp-15 (work in progress), July 2017.
- [I-D.ietf-core-multipart-ct]  
Fossati, T., Hartke, K., and C. Bormann, "Multipart  
Content-Format for CoAP", draft-ietf-core-multipart-ct-04  
(work in progress), August 2019.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119,  
DOI 10.17487/RFC2119, March 1997,  
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer  
Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,  
January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object  
Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049,  
October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC  
2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,  
May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert,  
"A Voucher Artifact for Bootstrapping Protocols",  
RFC 8366, DOI 10.17487/RFC8366, May 2018,  
<<https://www.rfc-editor.org/info/rfc8366>>.

## 13.2. Informative References

- [I-D.kumar-dice-dtls-relay]  
Kumar, S., Keoh, S., and O. Garcia-Morchon, "DTLS Relay for Constrained Environments", draft-kumar-dice-dtls-relay-02 (work in progress), October 2014.
- [I-D.richardson-anima-state-for-joinrouter]  
Richardson, M., "Considerations for stateful vs stateless join router in ANIMA bootstrap", draft-richardson-anima-state-for-joinrouter-03 (work in progress), September 2020.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<https://www.rfc-editor.org/info/rfc4944>>.
- [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <<https://www.rfc-editor.org/info/rfc6690>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<https://www.rfc-editor.org/info/rfc6775>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<https://www.rfc-editor.org/info/rfc7030>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/info/rfc7252>>.

## Appendix A. Stateless Proxy payload examples

The examples show the get coaps://[192.168.1.200]:5965/est/crts to a Registrar. The header generated between Client and registrar and from registrar to client are shown in detail. The DTLS encrypted code is not shown.

The request from Join Proxy to Registrar looks like:

```

85                                     # array(5)
  50                                 # bytes(16)
    000000000000000000000000FFFC0A801C8 #
  19 BDA7                             # unsigned(48551)
  0A                                   # unsigned(10)
  00                                   # unsigned(0)
  58 2D                               # bytes(45)
<cacrts DTLS encrypted request>

```

In CBOR Diagnostic:

```

[h'000000000000000000000000FFFC0A801C8', 48551, 10, 0,
 h'<cacrts DTLS encrypted request>']

```

The response is:

```

85                                     # array(5)
  50                                 # bytes(16)
    000000000000000000000000FFFC0A801C8 #
  19 BDA7                             # unsigned(48551)
  0A                                   # unsigned(10)
  00                                   # unsigned(0)
  59 026A                             # bytes(618)
<cacrts DTLS encrypted response>

```

In CBOR diagnostic:

```

[h'000000000000000000000000FFFC0A801C8', 48551, 10, 0,
 h'<cacrts DTLS encrypted response>']

```

## Authors' Addresses

Michael Richardson  
Sandelman Software Works

Email: mcr+ietf@sandelman.ca

Peter van der Stok  
vanderstok consultancy

Email: consultancy@vanderstok.org

Panos Kampanakis  
Cisco Systems

Email: pkampana@cisco.com

anima Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: 16 October 2022

M. Richardson  
Sandelman Software Works  
P. van der Stok  
vanderstok consultancy  
P. Kampanakis  
Cisco Systems  
14 April 2022

Constrained Join Proxy for Bootstrapping Protocols  
draft-ietf-anima-constrained-join-proxy-10

Abstract

This document extends the work of Bootstrapping Remote Secure Key Infrastructures (BRSKI) by replacing the Circuit-proxy between Pledge and Registrar by a stateless/stateful constrained Join Proxy. The constrained Join Proxy is a mesh neighbor of the Pledge and can relay a DTLS session originating from a Pledge with only link-local addresses to a Registrar which is not a mesh neighbor of the Pledge.

This document defines a protocol to securely assign a Pledge to a domain, represented by a Registrar, using an intermediary node between Pledge and Registrar. This intermediary node is known as a "constrained Join Proxy". An enrolled Pledge can act as a constrained Join Proxy.

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 16 October 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
2. Terminology . . . . .	5
3. Requirements Language . . . . .	5
4. constrained Join Proxy functionality . . . . .	5
5. constrained Join Proxy specification . . . . .	7
5.1. Stateful Join Proxy . . . . .	7
5.2. Stateless Join Proxy . . . . .	8
5.3. Stateless Message structure . . . . .	10
6. Discovery . . . . .	11
6.1. Join Proxy discovers Registrar . . . . .	12
6.1.1. CoAP discovery . . . . .	13
6.1.2. GRASP discovery . . . . .	13
6.1.3. 6tisch discovery . . . . .	13
6.2. Pledge discovers Registrar . . . . .	13
6.2.1. CoAP discovery . . . . .	13
6.2.2. GRASP discovery . . . . .	14
6.2.3. 6tisch discovery . . . . .	14
6.3. Pledge discovers Join Proxy . . . . .	14
6.3.1. CoAP discovery . . . . .	14
6.3.2. GRASP discovery . . . . .	15
6.3.3. 6tisch discovery . . . . .	15
7. Comparison of stateless and stateful modes . . . . .	15
8. Security Considerations . . . . .	16
9. IANA Considerations . . . . .	17
9.1. Resource Type Attributes registry . . . . .	17
9.2. service name and port number registry . . . . .	17
10. Acknowledgements . . . . .	18
11. Contributors . . . . .	18
12. Changelog . . . . .	18
12.1. 10 to 09 . . . . .	18
12.2. 09 to 07 . . . . .	18
12.3. 06 to 07 . . . . .	18
12.4. 05 to 06 . . . . .	19
12.5. 04 to 05 . . . . .	19
12.6. 03 to 04 . . . . .	19
12.7. 02 to 03 . . . . .	19
12.8. 01 to 02 . . . . .	19

12.9.	00 to 01 . . . . .	19
12.10.	00 to 00 . . . . .	19
13.	References . . . . .	19
13.1.	Normative References . . . . .	20
13.2.	Informative References . . . . .	21
Appendix A.	Stateless Proxy payload examples . . . . .	23
Authors' Addresses	. . . . .	23

## 1. Introduction

The Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol described in [RFC8995] provides a solution for a secure zero-touch (automated) bootstrap of new (unconfigured) devices. In the context of BRSKI, new devices, called "Pledges", are equipped with a factory-installed Initial Device Identifier (IDevID) (see [ieee802-1AR]), and are enrolled into a network. BRSKI makes use of Enrollment over Secure Transport (EST) [RFC7030] with [RFC8366] vouchers to securely enroll devices. A Registrar provides the security anchor of the network to which a Pledge enrolls. In this document, BRSKI is extended such that a Pledge connects to "Registrars" via a constrained Join Proxy. In particular, the underlying IP network is assumed to be a mesh network as described in [RFC4944], although other IP-over-foo networks are not excluded.

A complete specification of the terminology is pointed at in Section 2.

The specified solutions in [RFC8995] and [RFC7030] are based on POST or GET requests to the EST resources (/cacerts, /simpleenroll, /simplereenroll, /serverkeygen, and /csrattrs), and the brski resources (/requestvoucher, /voucher\_status, and /enrollstatus). These requests use https and may be too large in terms of code space or bandwidth required for constrained devices. Constrained devices which may be part of constrained networks [RFC7228], typically implement the IPv6 over Low-Power Wireless personal Area Networks (6LoWPAN) [RFC4944] and Constrained Application Protocol (CoAP) [RFC7252].

CoAP can be run with the Datagram Transport Layer Security (DTLS) [RFC6347] as a security protocol for authenticity and confidentiality of the messages. This is known as the "coaps" scheme. A constrained version of EST, using Coap and DTLS, is described in [I-D.ietf-ace-coap-est]. The [I-D.ietf-anima-constrained-voucher] extends [I-D.ietf-ace-coap-est] with BRSKI artifacts such as voucher, request voucher, and the protocol extensions for constrained Pledges.

DTLS is a client-server protocol relying on the underlying IP layer to perform the routing between the DTLS Client and the DTLS Server. However, the Pledge will not be IP routable over the mesh network until it is authenticated to the mesh network. A new Pledge can only initially use a link-local IPv6 address to communicate with a mesh neighbor [RFC6775] until it receives the necessary network configuration parameters. The Pledge receives these configuration parameters from the Registrar. When the Registrar is not a direct neighbor of the Registrar but several hops away, the Pledge discovers a neighbor constrained Join Proxy, which transmits the DTLS protected request coming from the Pledge to the Registrar. The constrained Join Proxy must be enrolled previously such that the message from constrained Join Proxy to Registrar can be routed over one or more hops.

During enrollment, a DTLS connection is required between Pledge and Registrar.

Once a Pledge is enrolled, it can act as constrained Join Proxy between other Pledges and the enrolling Registrar.

This document specifies a new form of constrained Join Proxy and protocol to act as intermediary between Pledge and Registrar to relay DTLS messages between Pledge and Registrar. Two modes of the constrained Join Proxy are specified:

- 1 A stateful Join Proxy that locally stores IP addresses during the connection.
- 2 A stateless Join Proxy that where the connection state is stored in the messages.

This document is very much inspired by text published earlier in [I-D.kumar-dice-dtls-relay].

[I-D.richardson-anima-state-for-joinrouter] outlined the various options for building a constrained Join Proxy. [RFC8995] adopted only the Circuit Proxy method (1), leaving the other methods as future work.

The stateful and stateless modes differ in the way that they store the state required to forward the return packet to the pledge. Similar to the difference between storing and non\_storing Modes of Operations (MOP) in RPL [RFC6550]. In the stateful method, the return forward state is stored in the join proxy. In the stateless method, the return forward state is stored in the network.

## 2. Terminology

The following terms are defined in [RFC8366], and are used identically as in that document: artifact, imprint, domain, Join Registrar/Coordinator (JRC), Pledge, and Voucher.

In this document, the term "Registrar" is used throughout instead of "Join Registrar/Coordinator (JRC)".

The term "installation network" refers to all devices in the installation and the network connections between them. The term "installation IP\_address" refers to an address out of the set of addresses which are routable over the whole installation network.

The "Constrained Join Proxy" enables a pledge that is multiple hops away from the Registrar, to securely execute the BRSKI protocol [RFC8995] over a secure channel.

The term "join Proxy" is used interchangeably with the term "constrained Join Proxy" throughout this document.

## 3. Requirements Language

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

## 4. constrained Join Proxy functionality

As depicted in the Figure 1, the Pledge (P), in a Low-Power and Lossy Network (LLN) mesh [RFC7102] can be more than one hop away from the Registrar (R) and not yet authenticated into the network.

In this situation, the Pledge can only communicate one-hop to its nearest neighbor, the constrained Join Proxy (J) using their link-local IPv6 addresses. However, the Pledge (P) needs to communicate with end-to-end security with a Registrar to authenticate and get the relevant system/network parameters. If the Pledge (P), knowing the IP-address of the Registrar, initiates a DTLS connection to the Registrar, then the packets are dropped at the constrained Join Proxy (J) since the Pledge (P) is not yet admitted to the network or there is no IP routability to Pledge (P) for any returned messages from the Registrar.

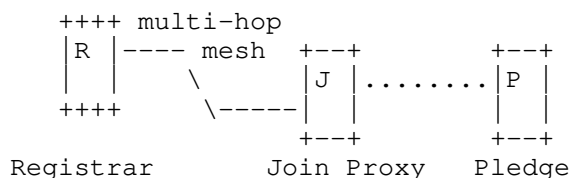


Figure 1: multi-hop enrollment.

Without routing the Pledge (P) cannot establish a secure connection to the Registrar (R) over multiple hops in the network.

Furthermore, the Pledge (P) cannot discover the IP address of the Registrar (R) over multiple hops to initiate a DTLS connection and perform authentication.

To overcome the problems with non-routability of DTLS packets and/or discovery of the destination address of the Registrar, the constrained Join Proxy is introduced. This constrained Join Proxy functionality is configured into all authenticated devices in the network which may act as a constrained Join Proxy for Pledges. The constrained Join Proxy allows for routing of the packets from the Pledge using IP routing to the intended Registrar. An authenticated constrained Join Proxy can discover the routable IP address of the Registrar over multiple hops. The following Section 5 specifies the two constrained Join Proxy modes. A comparison is presented in Section 7.

When a mesh network is set up, it consists of a Registrar and a set of connected pledges. No constrained Join Proxies are present. The wanted end-state is a network with a Registrar and a set of enrolled devices. Some of these enrolled devices can act as constrained Join Proxies. Pledges can only employ link-local communication until they are enrolled. A Pledge will regularly try to discover a constrained Join Proxy or a Registrar with link-local discovery requests. The Pledges which are neighbors of the Registrar will discover the Registrar and be enrolled following the BRSKI protocol. An enrolled device can act as constrained Join Proxy. The Pledges which are not a neighbor of the Registrar will eventually discover a constrained Join Proxy and follow the BRSKI protocol to be enrolled. While this goes on, more and more constrained Join Proxies with a larger hop distance to the Registrar will emerge. The network should be configured such that at the end of the enrollment process, all pledges have discovered a neighboring constrained Join Proxy or the Registrar, and all "legal" Pledges are enrolled.

## 5. constrained Join Proxy specification

A Join Proxy can operate in two modes:

- \* Stateful mode
- \* Stateless mode

A Join Proxy MAY implement both. A mechanism to switch between modes is out of scope of this document. It is recommended that a Join Proxy uses only one of these modes at any given moment during an installation lifetime.

### 5.1. Stateful Join Proxy

In stateful mode, the Join Proxy forwards the DTLS messages to the Registrar.

Assume that the Pledge does not know the IP address of the Registrar it needs to contact. The Join Proxy has been enrolled via the Registrar and learns the IP address and port of the Registrar, for example by using the discovery mechanism described in Section 6. The Pledge first discovers (see Section 6) and selects the most appropriate Join Proxy. (Discovery can also be based upon [RFC8995] section 4.1). For service discovery via DNS-SD [RFC6763], this document specifies the service names in Section 9.2. The Pledge initiates its request as if the Join Proxy is the intended Registrar. The Join Proxy receives the message at a discoverable join-port. The Join Proxy constructs an IP packet by copying the DTLS payload from the message received from the Pledge, and provides source and destination addresses to forward the message to the intended Registrar. The Join Proxy stores the 4-tuple array of the messages received from the Registrar and copies it back to the header of the message returned to the Pledge.

In Figure 2 the various steps of the message flow are shown, with 5684 being the standard coaps port:

Pledge (P)	Join Proxy (J)	Registrar (R)	Message	
			Src_IP:port	Dst_IP:port
--ClientHello-->			IP_P:p_P	IP_Jl:p_Jl
	--ClientHello-->		IP_Jr:p_Jr	IP_R:5684
		<--ServerHello--	IP_R:5684	IP_Jr:p_Jr
		:		
<--ServerHello--		:	IP_Jl:p_Jl	IP_P:p_P
	:	:		
[DTLS messages]		:	:	:
	:	:	:	:
--Finished-->		:	IP_P:p_P	IP_Jl:p_Jl
	--Finished-->		IP_Jr:p_Jr	IP_R:5684
		<--Finished--	IP_R:5684	IP_Jr:p_Jr
<--Finished--			IP_Jl:p_Jl	IP_P:p_P
:		:	:	:

IP\_P:p\_P = Link-local IP address and port of Pledge (DTLS Client)

IP\_R:5684 = Routable IP address and coaps port of Registrar

IP\_J1:p\_J1 = Link-local IP address and join-port of Join Proxy

IP\_Jr:p\_Jr = Routable IP address and client port of Join Proxy

Figure 2: constrained stateful joining message flow with Registrar address known to Join Proxy.

## 5.2. Stateless Join Proxy

The stateless Join Proxy aims to minimize the requirements on the constrained Join Proxy device. Stateless operation requires no memory in the Join Proxy device, but may also reduce the CPU impact as the device does not need to search through a state table.

If an untrusted Pledge that can only use link-local addressing wants to contact a trusted Registrar, and the Registrar is more than one hop away, it sends its DTLS messages to the Join Proxy.

When a Pledge attempts a DTLS connection to the Join Proxy, it uses its link-local IP address as its IP source address. This message is transmitted one-hop to a neighboring (Join Proxy) node. Under normal circumstances, this message would be dropped at the neighbor node since the Pledge is not yet IP routable or is not yet authenticated to send messages through the network. However, if the neighbor device has the Join Proxy functionality enabled; it routes the DTLS message to its Registrar of choice.

The Join Proxy transforms the DTLS message to a JPY message which includes the DTLS data as payload, and sends the JPY message to the join-port of the Registrar.

The JPY message payload consists of two parts:

- \* Header (H) field: consisting of the source link-local address and port of the Pledge (P), and
- \* Contents (C) field: containing the original DTLS payload.

On receiving the JPY message, the Registrar (or proxy) retrieves the two parts.

The Registrar transiently stores the Header field information. The Registrar uses the Contents field to execute the Registrar functionality. However, when the Registrar replies, it also extends its DTLS message with the header field in a JPY message and sends it back to the Join Proxy. The Registrar SHOULD NOT assume that it can decode the Header Field, it should simply repeat it when responding. The Header contains the original source link-local address and port of the Pledge from the transient state stored earlier and the Contents field contains the DTLS payload.

On receiving the JPY message, the Join Proxy retrieves the two parts. It uses the Header field to route the DTLS message containing the DTLS payload retrieved from the Contents field to the Pledge.

In this scenario, both the Registrar and the Join Proxy use discoverable join-ports, for the Join Proxy this may be a default CoAP port.

The Figure 3 depicts the message flow diagram:

Pledge (P)	Join Proxy (J)	Registrar (R)	Message	
			Src_IP:port	Dst_IP:port
<pre> --ClientHello--&gt;     --JPY[H(IP_P:p_P),--&gt;         C(ClientHello)]     &lt;--JPY[H(IP_P:p_P),--         C(ServerHello)]  &lt;--ServerHello-- : [ DTLS messages ] : --Finished--&gt;     --JPY[H(IP_P:p_P),--&gt;         C(Finished)]     &lt;--JPY[H(IP_P:p_P),--         C(Finished)]  &lt;--Finished-- : </pre>			IP_P:p_P	IP_Jl:p_Jl
			IP_Jr:p_Jr	IP_R:p_Ra
			IP_R:p_Ra	IP_Jr:p_Jr
			IP_Jl:p_Jl	IP_P:p_P
			:	:
			:	:
			IP_P:p_P	IP_Jr:p_Jr
			IP_Jl:p_Jl	IP_R:p_Ra
			IP_R:p_Ra	IP_Jr:p_Jr
			IP_Jl:p_Jl	IP_P:p_P
			:	:

IP\_P:p\_P = Link-local IP address and port of the Pledge

IP\_R:p\_Ra = Routable IP address and join-port of Registrar

IP\_Jl:p\_Jl = Link-local IP address and join-port of Join Proxy

IP\_Jr:p\_Jr = Routable IP address and port of Join Proxy

JPY[H()],C()) = Join Proxy message with header H and content C

Figure 3: constrained stateless joining message flow.

### 5.3. Stateless Message structure

The JPY message is constructed as a payload with media-type application/cbor

Header and Contents fields together are one CBOR array of 5 elements:

1. header field: containing a CBOR array [RFC8949] with the Pledge IPv6 Link Local address as a CBOR byte string, the Pledge's UDP port number as a CBOR integer, the IP address family (IPv4/IPv6) as a CBOR integer, and the proxy's ifindex or other identifier for the physical port as CBOR integer. The header field is not DTLS encrypted.
2. Content field: containing the DTLS payload as a CBOR byte string.

The address family integer is defined in [family] with:

- 1 IP (IP version 4)
- 2 IP6 (IP version 6)

The Join Proxy cannot decrypt the DTLS payload and has no knowledge of the transported media type.

```
JPY_message =  
[  
  ip      : bstr,  
  port    : int,  
  family  : int,  
  index   : int  
  content : bstr  
]
```

Figure 4: CDDL representation of JPY message

The contents are DTLS encrypted. In CBOR diagnostic notation the payload JPY[H(IP\_P:p\_P)], will look like:

```
[h'IP_p', p_P, family, ident, h'DTLS-payload']
```

On reception by the Registrar, the Registrar MUST verify that the number of array elements is larger than or equal to 5, and reject the message when the number of array elements is smaller than 5. After replacing the 5th "content" element with the DTLS payload of the response message and leaving all other array elements unchanged, the Registrar returns the response message.

Examples are shown in Appendix A.

The header field is completely opaque to the receiver. A Registrar MUST copy the header and return it unmodified in the return message.

## 6. Discovery

It is assumed that Join Proxy seamlessly provides a coaps connection between Pledge and Registrar. In particular this section extends section 4.1 of [RFC8995] for the constrained case.

The discovery follows two steps with two alternatives for step 1:

- \* Step 1. Two alternatives exist (near and remote):

- Near: the Pledge is one hop away from the Registrar. The Pledge discovers the link-local address of the Registrar as described in [I-D.ietf-ace-coap-est]. From then on, it follows the BRSKI process as described in [I-D.ietf-ace-coap-est] and [I-D.ietf-anima-constrained-voucher], using link-local addresses.
  - Remote: the Pledge is more than one hop away from a relevant Registrar, and discovers the link-local address and join-port of a Join Proxy. The Pledge then follows the BRSKI procedure using the link-local address of the Join Proxy.
- \* Step 2. The enrolled Join Proxy discovers the join-port of the Registrar.

The order in which the two alternatives of step 1 are tried is installation dependent. The trigger for discovery in Step 2 is implementation dependent.

Once a Pledge is enrolled, it may function as Join Proxy. The Join Proxy functions are advertised as described below. In principle, the Join Proxy functions are offered via a join-port, and not the standard coaps port. Also, the Registrar offers a join-port to which the stateless Join Proxy sends the JPY message. The Join Proxy and Registrar show the extra join-port number when responding to a /.well-known/core discovery request addressed to the standard coap/coaps port.

Three discovery cases are discussed: Join Proxy discovers Registrar, Pledge discovers Registrar, and Pledge discovers Join Proxy. Each discovery case considers three alternatives: CoAP based discovery, GRASP Based discovery, and 6tisch based discovery. The choice of discovery mechanism depends on the type of installation, and manufacturers can provide the pledge/Join Proxy with support for more than one discovery mechanism. The pledge/Join Proxy can be designed to dynamically try different discovery mechanisms until a successful discovery mechanism is found, or the choice of discovery mechanism could be configured during device installation.

#### 6.1. Join Proxy discovers Registrar

In this section, the Join Proxy and Registrar are assumed to communicate via Link-Local addresses. This section describes the discovery of the Registrar by the Join Proxy.

#### 6.1.1. CoAP discovery

The discovery of the coaps Registrar, using coap discovery, by the Join Proxy follows sections 6.3 and 6.5.1 of [I-D.ietf-anima-constrained-voucher]. The stateless Join Proxy can discover the join-port of the Registrar by sending a GET request to `"/.well-known/core"` including a resource type (rt) parameter with the value `"brski.rjp"` [RFC6690]. Upon success, the return payload will contain the join-port of the Registrar.

```
REQ: GET coap://[IP_address]/.well-known/core?rt=brski.rjp
```

```
RES: 2.05 Content
```

```
<coaps://[IP_address]:join-port>; rt="brski.rjp"
```

The discoverable port numbers are usually returned for Join Proxy resources in the `<URI-Reference>` of the payload (see section 5.1 of [I-D.ietf-ace-coap-est]).

#### 6.1.2. GRASP discovery

This section is normative for uses with an ANIMA ACP. In the context of autonomic networks, the Join Proxy uses the DULL GRASP M\_FLOOD mechanism to announce itself. Section 4.1.1 of [RFC8995] discusses this in more detail. The Registrar announces itself using ACP instance of GRASP using M\_FLOOD messages. Autonomic Network Join Proxies MUST support GRASP discovery of Registrar as described in section 4.3 of [RFC8995].

#### 6.1.3. 6tisch discovery

The discovery of the Registrar by the Join Proxy uses the enhanced beacons as discussed in [I-D.ietf-6tisch-enrollment-enhanced-beacon].

### 6.2. Pledge discovers Registrar

In this section, the Pledge and Registrar are assumed to communicate via Link-Local addresses. This section describes the discovery of the Registrar by the Pledge.

#### 6.2.1. CoAP discovery

The discovery of the coaps Registrar, using coap discovery, by the Pledge follows sections 6.3 and 6.5.1 of [I-D.ietf-anima-constrained-voucher].

### 6.2.2. GRASP discovery

This section is normative for uses with an ANIMA ACP. In the context of autonomic networks, the Pledge uses the DULL GRASP M\_FLOOD mechanism to announce itself. Section 4.1.1 of [RFC8995] discusses this in more detail. The Registrar announces itself using ACP instance of GRASP using M\_FLOOD messages. Autonomic Network Join Proxies MUST support GRASP discovery of Registrar as described in section 4.3 of [RFC8995] .

### 6.2.3. 6tisch discovery

The discovery of Registrar by the Pledge uses the enhanced beacons as discussed in [I-D.ietf-6tisch-enrollment-enhanced-beacon].

## 6.3. Pledge discovers Join Proxy

In this section, the Pledge and Join Proxy are assumed to communicate via Link-Local addresses. This section describes the discovery of the Join Proxy by the Pledge.

### 6.3.1. CoAP discovery

In the context of a coap network without Autonomic Network support, discovery follows the standard coap policy. The Pledge can discover a Join Proxy by sending a link-local multicast message to ALL CoAP Nodes with address FF02::FD. Multiple or no nodes may respond. The handling of multiple responses and the absence of responses follow section 4 of [RFC8995].

The join-port of the Join Proxy is discovered by sending a GET request to `"/.well-known/core"` including a resource type (rt) parameter with the value `"brski.jp"` [RFC6690]. Upon success, the return payload will contain the join-port.

The example below shows the discovery of the join-port of the Join Proxy.

```
REQ: GET coap://[FF02::FD]/.well-known/core?rt=brski.jp
```

```
RES: 2.05 Content
```

```
<coaps://[IP_address]:join-port>; rt="brski.jp"
```

Port numbers are assumed to be the default numbers 5683 and 5684 for coap and coaps respectively (sections 12.6 and 12.7 of [RFC7252]) when not shown in the response. Discoverable port numbers are usually returned for Join Proxy resources in the `<URI-Reference>` of the payload (see section 5.1 of [I-D.ietf-ace-coap-est]).

### 6.3.2. GRASP discovery

This section is normative for uses with an ANIMA ACP. The Pledge MUST listen for GRASP M\_FLOOD [RFC8990] announcements of the objective: "AN\_Proxy". See section 4.1.1 [RFC8995] for the details of the objective.

### 6.3.3. 6tisch discovery

The discovery of the Join Proxy by the Pledge uses the enhanced beacons as discussed in [I-D.ietf-6tisch-enrollment-enhanced-beacon].

## 7. Comparison of stateless and stateful modes

The stateful and stateless mode of operation for the Join Proxy have their advantages and disadvantages. This section should enable to make a choice between the two modes based on the available device resources and network bandwidth.

Properties	Stateful mode	Stateless mode
State Information	The Join Proxy needs additional storage to maintain mapping between the address and port number of the Pledge and those of the Registrar.	No information is maintained by the Join Proxy. Registrar needs to store the packet header.
Packet size	The size of the forwarded message is the same as the original message.	Size of the forwarded message is bigger than the original, it includes additional information
Specification complexity	The Join Proxy needs additional functionality to maintain state information, and specify the source and destination addresses of the DTLS handshake messages	New JPY message to encapsulate DTLS payload The Registrar and the Join Proxy have to understand the JPY message in order to process it.
Ports	Join Proxy needs discoverable join-port	Join Proxy and Registrar need discoverable join-ports

Figure 5: Comparison between stateful and stateless mode

## 8. Security Considerations

All the concerns in [RFC8995] section 4.1 apply. The Pledge can be deceived by malicious Join Proxy announcements. The Pledge will only join a network to which it receives a valid [RFC8366] voucher [I-D.ietf-anima-constrained-voucher]. Once the Pledge joined, the payload between Pledge and Registrar is protected by DTLS.

A malicious constrained Join Proxy has a number of routing possibilities:

- \* It sends the message on to a malicious Registrar. This is the same case as the presence of a malicious Registrar discussed in RFC 8995.
- \* It does not send on the request or does not return the response from the Registrar. This is the case of the not responding or crashing Registrar discussed in RFC 8995.
- \* It uses the returned response of the Registrar to enroll itself in the network. With very low probability it can decrypt the response. Successful enrollment is deemed too unlikely.
- \* It uses the request from the pledge to appropriate the pledge certificate, but then it still needs to acquire the private key of the pledge. Also this is assumed to be highly unlikely.
- \* A malicious node can construct an invalid Join Proxy message. Suppose, the destination port is the coaps port. In that case, a Join Proxy can accept the message and add the routing addresses without checking the payload. The Join Proxy then routes it to the Registrar. In all cases, the Registrar needs to receive the message at the join-port, checks that the message consists of two parts and uses the DTLS payload to start the BRSKI procedure. It is highly unlikely that this malicious payload will lead to node acceptance.
- \* A malicious node can sniff the messages routed by the constrained Join Proxy. It is very unlikely that the malicious node can decrypt the DTLS payload. A malicious node can read the header field of the message sent by the stateless Join Proxy. This ability does not yield much more information than the visible addresses transported in the network packets.

It should be noted here that the contents of the CBOR array used to convey return address information is not DTLS protected. When the communication between JOIN Proxy and Registrar passes over an unsecure network, an attacker can change the CBOR array, causing the

Registrar to deviate traffic from the intended Pledge. These concerns are also expressed in [RFC8974]. It is also pointed out that the encryption in the source is a local matter. Similarly to [RFC8974], the use of AES-CCM [RFC3610] with a 64-bit tag is recommended, combined with a sequence number and a replay window.

If such scenario needs to be avoided, the constrained Join Proxy MUST encrypt the CBOR array using a locally generated symmetric key. The Registrar is not able to examine the encrypted result, but does not need to. The Registrar stores the encrypted header in the return packet without modifications. The constrained Join Proxy can decrypt the contents to route the message to the right destination.

In some installations, layer 2 protection is provided between all member pairs of the mesh. In such an environment encryption of the CBOR array is unnecessary because the layer 2 protection already provide it.

## 9. IANA Considerations

### 9.1. Resource Type Attributes registry

This specification registers two new Resource Type (rt=) Link Target Attributes in the "Resource Type (rt=) Link Target Attribute Values" subregistry under the "Constrained RESTful Environments (CoRE) Parameters" registry per the [RFC6690] procedure.

Attribute Value: brski.jp

Description: This BRSKI resource type is used to query and return the supported BRSKI resources of the constrained Join Proxy.

Reference: [this document]

Attribute Value: brski.rjp

Description: This BRSKI resource type is used for the constrained Join Proxy to query and return Join Proxy specific BRSKI resources of a Registrar.

Reference: [this document]

### 9.2. service name and port number registry

This specification registers two service names under the "Service Name and Transport Protocol Port Number" registry.

Service Name: brski-jp  
Transport Protocol(s): udp  
Assignee: IESG <iesg@ietf.org>  
Contact: IESG <iesg@ietf.org>  
Description: Bootstrapping Remote Secure Key Infrastructure  
              constrained Join Proxy  
Reference: [this document]

Service Name: brski-rjp  
Transport Protocol(s): udp  
Assignee: IESG <iesg@ietf.org>  
Contact: IESG <iesg@ietf.org>  
Description: Bootstrapping Remote Secure Key Infrastructure  
              Registrar join-port used by stateless constrained  
              Join Proxy  
Reference: [this document]

## 10. Acknowledgements

Many thanks for the comments by Carlsen, Bormann, Brian Carpenter, Esko Dijk, Toerless Eckert, Russ Housley, Ines Robles, Juergen Schoenwaelder, Malisa Vu&#269;ini&#263;, and Rob Wilton.

## 11. Contributors

Sandeep Kumar, Sye loong Keoh, and Oscar Garcia-Morchon are the co-authors of the draft-kumar-dice-dtls-relay-02. Their draft has served as a basis for this document. Much text from their draft is copied over to this draft.

## 12. Changelog

### 12.1. 10 to 09

- \* OPSDIR review
- \* IANA review
- \* SECDIR review
- \* GENART review

### 12.2. 09 to 07

- \* typos

### 12.3. 06 to 07

- \* AD review changes

## 12.4. 05 to 06

- \* RT value change to brski.jp and brski.rjp
- \* new registry values for IANA
- \* improved handling of jpy header array

## 12.5. 04 to 05

- \* Join Proxy and join-port consistent spelling
- \* some nits removed
- \* restructured discovery section
- \* rephrased parts of security section

## 12.6. 03 to 04

- \* mail address and reference

## 12.7. 02 to 03

- \* Terminology updated
- \* Several clarifications on discovery and routability
- \* DTLS payload introduced

## 12.8. 01 to 02

- \* Discovery of Join Proxy and Registrar ports

## 12.9. 00 to 01

- \* Registrar used throughout instead of EST server
- \* Emphasized additional Join Proxy port for Join Proxy and Registrar
- \* updated discovery accordingly
- \* updated stateless Join Proxy JPY header
- \* JPY header described with CDDL
- \* Example simplified and corrected

## 12.10. 00 to 00

- \* copied from vanderstok-anima-constrained-join-proxy-05

## 13. References

## 13.1. Normative References

- [family] "Address Family Numbers", 19 October 2021,  
<[https://www.iana.org/assignments/address-family-numbers/  
address-family-numbers.xhtml](https://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml)>.
- [I-D.ietf-ace-coap-est]  
Stok, P. V. D., Kampanakis, P., Richardson, M. C., and S.  
Raza, "EST over secure CoAP (EST-coaps)", Work in  
Progress, Internet-Draft, draft-ietf-ace-coap-est-18, 6  
January 2020, <[https://www.ietf.org/archive/id/draft-ietf-  
ace-coap-est-18.txt](https://www.ietf.org/archive/id/draft-ietf-ace-coap-est-18.txt)>.
- [I-D.ietf-anima-constrained-voucher]  
Richardson, M., Stok, P. V. D., Kampanakis, P., and E.  
Dijk, "Constrained Bootstrapping Remote Secure Key  
Infrastructure (BRSKI)", Work in Progress, Internet-Draft,  
draft-ietf-anima-constrained-voucher-17, 7 April 2022,  
<[https://www.ietf.org/archive/id/draft-ietf-anima-  
constrained-voucher-17.txt](https://www.ietf.org/archive/id/draft-ietf-anima-<br/>constrained-voucher-17.txt)>.
- [ieee802-1AR]  
"IEEE 802.1AR Secure Device Identifier", 2009,  
<<https://standards.ieee.org/standard/802.1AR-2009.html>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate  
Requirement Levels", BCP 14, RFC 2119,  
DOI 10.17487/RFC2119, March 1997,  
<<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer  
Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,  
January 2012, <<https://www.rfc-editor.org/info/rfc6347>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC  
2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,  
May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert,  
"A Voucher Artifact for Bootstrapping Protocols",  
RFC 8366, DOI 10.17487/RFC8366, May 2018,  
<<https://www.rfc-editor.org/info/rfc8366>>.
- [RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object  
Representation (CBOR)", STD 94, RFC 8949,  
DOI 10.17487/RFC8949, December 2020,  
<<https://www.rfc-editor.org/info/rfc8949>>.

- [RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <<https://www.rfc-editor.org/info/rfc8990>>.
- [RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <<https://www.rfc-editor.org/info/rfc8995>>.

### 13.2. Informative References

- [I-D.ietf-6tisch-enrollment-enhanced-beacon]  
(editor), D. D. and M. Richardson, "Encapsulation of 6TiSCH Join and Enrollment Information Elements", Work in Progress, Internet-Draft, draft-ietf-6tisch-enrollment-enhanced-beacon-14, 21 February 2020, <<https://www.ietf.org/archive/id/draft-ietf-6tisch-enrollment-enhanced-beacon-14.txt>>.
- [I-D.kumar-dice-dtls-relay]  
Kumar, S. S., Keoh, S. L., and O. Garcia-Morchon, "DTLS Relay for Constrained Environments", Work in Progress, Internet-Draft, draft-kumar-dice-dtls-relay-02, 20 October 2014, <<https://www.ietf.org/archive/id/draft-kumar-dice-dtls-relay-02.txt>>.
- [I-D.richardson-anima-state-for-joinrouter]  
Richardson, M. C., "Considerations for stateful vs stateless join router in ANIMA bootstrap", Work in Progress, Internet-Draft, draft-richardson-anima-state-for-joinrouter-03, 22 September 2020, <<https://www.ietf.org/archive/id/draft-richardson-anima-state-for-joinrouter-03.txt>>.
- [RFC3610] Whiting, D., Housley, R., and N. Ferguson, "Counter with CBC-MAC (CCM)", RFC 3610, DOI 10.17487/RFC3610, September 2003, <<https://www.rfc-editor.org/info/rfc3610>>.
- [RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<https://www.rfc-editor.org/info/rfc4944>>.

- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <<https://www.rfc-editor.org/info/rfc6690>>.
- [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.
- [RFC6775] Shelby, Z., Ed., Chakrabarti, S., Nordmark, E., and C. Bormann, "Neighbor Discovery Optimization for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 6775, DOI 10.17487/RFC6775, November 2012, <<https://www.rfc-editor.org/info/rfc6775>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <<https://www.rfc-editor.org/info/rfc7030>>.
- [RFC7102] Vasseur, JP., "Terms Used in Routing for Low-Power and Lossy Networks", RFC 7102, DOI 10.17487/RFC7102, January 2014, <<https://www.rfc-editor.org/info/rfc7102>>.
- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.
- [RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <<https://www.rfc-editor.org/info/rfc7252>>.
- [RFC8974] Hartke, K. and M. Richardson, "Extended Tokens and Stateless Clients in the Constrained Application Protocol (CoAP)", RFC 8974, DOI 10.17487/RFC8974, January 2021, <<https://www.rfc-editor.org/info/rfc8974>>.

## Appendix A. Stateless Proxy payload examples

The examples show the request "GET coaps://192.168.1.200:5965/est/crts" to a Registrar. The header generated between Join Proxy and Registrar and from Registrar to Join Proxy are shown in detail. The DTLS payload is not shown.

The request from Join Proxy to Registrar looks like:

```

85                                     # array(5)
  50                                 # bytes(16)
    FE800000000000000000000000000000FFFFC0A801C8 #
  19 BDA7                             # unsigned(48551)
  01                                 # unsigned(1) IP
  00                                 # unsigned(0)
  58 2D                             # bytes(45)
<cacrts DTLS encrypted request>

```

In CBOR Diagnostic:

```

[h'FE800000000000000000000000000000FFFFC0A801C8', 48551, 1, 0,
 h'<cacrts DTLS encrypted request>']

```

The response is:

```

85                                     # array(5)
  50                                 # bytes(16)
    FE800000000000000000000000000000FFFFC0A801C8 #
  19 BDA7                             # unsigned(48551)
  01                                 # unsigned(1) IP
  00                                 # unsigned(0)
  59 026A                           # bytes(618)
<cacrts DTLS encrypted response>

```

In CBOR diagnostic:

```

[h'FE800000000000000000000000000000FFFFC0A801C8', 48551, 1, 0,
 h'<cacrts DTLS encrypted response>']

```

## Authors' Addresses

Michael Richardson  
 Sandelman Software Works  
 Email: mcr+ietf@sandelman.ca

Peter van der Stok  
 vanderstok consultancy

Email: [stokcons@bbhmail.nl](mailto:stokcons@bbhmail.nl)

Panos Kampanakis  
Cisco Systems  
Email: [pkampana@cisco.com](mailto:pkampana@cisco.com)

anima Working Group  
Internet-Draft  
Updates: 8366, 8995 (if approved)  
Intended status: Standards Track  
Expires: 9 October 2022

M. Richardson  
Sandelman Software Works  
P. van der Stok  
vanderstok consultancy  
P. Kampanakis  
Cisco Systems  
E. Dijk  
IoTconsultancy.nl  
7 April 2022

Constrained Bootstrapping Remote Secure Key Infrastructure (BRSKI)  
draft-ietf-anima-constrained-voucher-17

Abstract

This document defines the Constrained Bootstrapping Remote Secure Key Infrastructure (Constrained BRSKI) protocol, which provides a solution for secure zero-touch bootstrapping of resource-constrained (IoT) devices into the network of a domain owner. This protocol is designed for constrained networks, which may have limited data throughput or may experience frequent packet loss. Constrained BRSKI is a variant of the BRSKI protocol, which uses an artifact signed by the device manufacturer called the "voucher" which enables a new device and the owner's network to mutually authenticate. While the BRSKI voucher is typically encoded in JSON, Constrained BRSKI defines a compact CBOR-encoded voucher. The BRSKI voucher is extended with new data types that allow for smaller voucher sizes. The Enrollment over Secure Transport (EST) protocol, used in BRSKI, is replaced with EST-over-CoAPS; and HTTPS used in BRSKI is replaced with CoAPS.

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 9 October 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	4
2. Terminology . . . . .	5
3. Requirements Language . . . . .	6
4. Overview of Protocol . . . . .	6
5. Updates to RFC8366 and RFC8995 . . . . .	7
6. BRSKI-EST Protocol . . . . .	8
6.1. Registrar and the Server Name Indicator (SNI) . . . . .	8
6.2. TLS Client Certificates: IDevID authentication . . . . .	9
6.3. Discovery, URIs and Content Formats . . . . .	10
6.3.1. RFC8995 Telemetry Returns . . . . .	12
6.4. Join Proxy options . . . . .	13
6.5. Extensions to BRSKI . . . . .	13
6.5.1. Discovery . . . . .	13
6.5.2. CoAP responses . . . . .	13
6.6. Extensions to EST-coaps . . . . .	14
6.6.1. Pledge Extensions . . . . .	15
6.6.2. EST-client Extensions . . . . .	16
6.6.3. Registrar Extensions . . . . .	18
6.7. DTLS handshake fragmentation Considerations . . . . .	19
7. BRSKI-MASA Protocol . . . . .	19
7.1. Protocol and Formats . . . . .	19
7.2. Registrar Voucher Request . . . . .	20
7.3. MASA and the Server Name Indicator (SNI) . . . . .	20
7.3.1. Registrar Certificate Requirement . . . . .	21
8. Pinning in Voucher Artifacts . . . . .	21
8.1. Registrar Identity Selection and Encoding . . . . .	21
8.2. MASA Pinning Policy . . . . .	23
8.3. Pinning of Raw Public Keys . . . . .	24
8.4. Considerations for use of IDevID-Issuer . . . . .	25
9. Artifacts . . . . .	26
9.1. Voucher Request artifact . . . . .	26
9.1.1. Tree Diagram . . . . .	26

9.1.2.	SID values . . . . .	27
9.1.3.	YANG Module . . . . .	28
9.1.4.	Example voucher request artifact . . . . .	32
9.2.	Voucher artifact . . . . .	32
9.2.1.	Tree Diagram . . . . .	32
9.2.2.	SID values . . . . .	33
9.2.3.	YANG Module . . . . .	33
9.2.4.	Example voucher artifacts . . . . .	36
9.3.	Signing voucher and voucher-request artifacts with COSE . . . . .	37
10.	Deployment-specific Discovery Considerations . . . . .	38
10.1.	6TSCH Deployments . . . . .	39
10.2.	Generic networks using GRASP . . . . .	39
10.3.	Generic networks using mDNS . . . . .	39
10.4.	Thread networks using Mesh Link Establishment (MLE) . . . . .	39
10.5.	Non-mesh networks using CoAP Discovery . . . . .	40
11.	Design Considerations . . . . .	40
12.	Raw Public Key Use Considerations . . . . .	40
12.1.	The Registrar Trust Anchor . . . . .	40
12.2.	The Pledge Voucher Request . . . . .	41
12.3.	The Voucher Response . . . . .	41
13.	Use of constrained vouchers with HTTPS . . . . .	41
14.	Security Considerations . . . . .	42
14.1.	Duplicate serial-numbers . . . . .	42
14.2.	IDevID security in Pledge . . . . .	43
14.3.	Security of CoAP and UDP protocols . . . . .	44
14.4.	Registrar Certificate may be self-signed . . . . .	45
14.5.	Use of RPK alternatives to proximity-registrar-cert . . . . .	45
14.6.	MASA support of CoAPS . . . . .	46
15.	IANA Considerations . . . . .	46
15.1.	Resource Type Registry . . . . .	46
15.2.	The IETF XML Registry . . . . .	47
15.3.	The YANG Module Names Registry . . . . .	47
15.4.	The RFC SID range assignment sub-registry . . . . .	47
15.5.	Media Types Registry . . . . .	48
15.5.1.	application/voucher-cose+cbor . . . . .	48
15.6.	CoAP Content-Format Registry . . . . .	48
15.7.	Update to BRSKI Parameters Registry . . . . .	49
16.	Acknowledgements . . . . .	49
17.	Changelog . . . . .	50
18.	References . . . . .	50
18.1.	Normative References . . . . .	50
18.2.	Informative References . . . . .	54
Appendix A.	Library Support for BRSKI . . . . .	56
A.1.	OpenSSL . . . . .	57
A.2.	mbedtls . . . . .	58
Appendix B.	Constrained BRSKI-EST Message Examples . . . . .	59
B.1.	enrollstatus . . . . .	59

B.2. voucher_status . . . . .	60
Appendix C. COSE-signed Voucher (Request) Examples . . . . .	61
C.1. Pledge, Registrar and MASA Keys . . . . .	61
C.1.1. Pledge IDevID private key . . . . .	61
C.1.2. Registrar private key . . . . .	61
C.1.3. MASA private key . . . . .	62
C.2. Pledge, Registrar and MASA Certificates . . . . .	62
C.2.1. Pledge IDevID Certificate . . . . .	62
C.2.2. Registrar Certificate . . . . .	64
C.2.3. MASA Certificate . . . . .	66
C.3. COSE-signed Pledge Voucher Request (PVR) . . . . .	68
C.4. COSE-signed Registrar Voucher Request (RVR) . . . . .	70
C.5. COSE-signed Voucher from MASA . . . . .	72
Appendix D. Generating Certificates with OpenSSL . . . . .	74
Appendix E. Pledge Device Class Profiles . . . . .	77
E.1. Minimal Pledge . . . . .	78
E.2. Typical Pledge . . . . .	78
E.3. Full-featured Pledge . . . . .	78
E.4. Comparison Chart of Pledge Classes . . . . .	78
Contributors . . . . .	79
Authors' Addresses . . . . .	80

## 1. Introduction

Secure enrollment of new nodes into constrained networks with constrained nodes presents unique challenges. As explained in [RFC7228], the networks are challenged and the nodes are constrained by energy, memory space, and code size.

The Bootstrapping Remote Secure Key Infrastructure (BRSKI) protocol described in [RFC8995] provides a solution for secure zero-touch (automated) bootstrap of new (unconfigured) devices. In it, new devices, such as IoT devices, are called "pledges", and equipped with a factory-installed Initial Device Identifier (IDevID) (see [ieee802-1AR]), are enrolled into a network.

The BRSKI solution described in [RFC8995] was designed to be modular, and this document describes a version scaled to the constraints of IoT deployments.

Therefore, this document defines a constrained version of the voucher artifact (described in [RFC8366]), along with a constrained version of BRSKI. This constrained-BRSKI protocol makes use of the constrained CoAP-based version of EST (EST-coaps from [I-D.ietf-ace-coap-est]) rather than the EST over HTTPS [RFC7030]. Constrained-BRSKI is itself scalable to multiple resource levels through the definition of optional functions. Appendix E illustrates this.

In BRSKI, the [RFC8366] voucher is by default serialized to JSON with a signature in CMS [RFC5652]. This document defines a new voucher serialization to CBOR [RFC8949] with a signature in COSE [I-D.ietf-cose-rfc8152bis-struct].

This COSE-signed CBOR-encoded voucher is transported using both secured CoAP and HTTPS. The CoAP connection (between Pledge and Registrar) is to be protected by either OSCORE+EDHOC [I-D.ietf-lake-edhoc] or DTLS (CoAPS). The HTTP connection (between Registrar and MASA) is to be protected using TLS (HTTPS).

This document specifies a constrained voucher-request artifact based on Section 3 of [RFC8995], and voucher(-request) transport over CoAP based on Section 3 of [RFC8995] and on [I-D.ietf-ace-coap-est].

The CBOR definitions for the constrained voucher format are defined using the mechanism described in [I-D.ietf-core-yang-cbor] using the SID mechanism explained in [I-D.ietf-core-sid]. As the tooling to convert YANG documents into a list of SID keys is still in its infancy, the table of SID values presented here MUST be considered normative rather than the output of the tool specified in [I-D.ietf-core-sid].

## 2. Terminology

The following terms are defined in [RFC8366], and are used identically as in that document: artifact, domain, imprint, Join Registrar/Coordinator (JRC), Manufacturer Authorized Signing Authority (MASA), Pledge, Registrar, Trust of First Use (TOFU), and Voucher.

The following terms from [RFC8995] are used identically as in that document: Domain CA, enrollment, IDevID, Join Proxy, LDevID, manufacturer, nonced, nonceless, PKIX.

The term Pledge Voucher Request, or acronym PVR, is introduced to refer to the voucher request between the pledge and the Registrar.

The term Registrar Voucher Request, or acronym RVR, is introduced to refer to the voucher request between the Registrar and the MASA.

In code examples, the string "<CODE BEGINS>" denotes the start of a code example and "<CODE ENDS>" the end of the code example. Four dots ("....") in a CBOR diagnostic notation byte string denotes a further sequence of bytes that is not shown for brevity.

### 3. Requirements Language

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

### 4. Overview of Protocol

[RFC8366] provides for vouchers that assert proximity, authenticate the Registrar, and can offer varying levels of anti-replay protection.

The proximity proof provided for in [RFC8366], is an assertion that the Pledge and the Registrar are believed to be close together, from a network topology point of view. Like in [RFC8995], proximity is shown by making TLS connections between the Pledge and Registrar using IPv6 Link-Local addresses.

The TLS connection is used to make a Voucher Request. This request is verified by an agent of the Pledge's manufacturer, which then issues a voucher. The voucher provides an authorization statement from the manufacturer indicating that the Registrar is the intended owner of the device. The voucher refers to the Registrar through pinning of the Registrar's identity.

This document does not make any extensions to the semantic meaning of vouchers, only the encoding has been changed to optimize for constrained devices and networks. The two main parts of the BRSKI protocol are named separately in this document: BRSKI-EST for the protocol between Pledge and Registrar, and BRSKI-MASA for the protocol between the Registrar and the MASA.

Time-based vouchers are supported in this definition, but given that constrained devices are extremely unlikely to have accurate time, their use will be uncommon. Most Pledges using constrained vouchers will be online during enrollment and will use live nonces to provide anti-replay protection rather than expiry times.

[RFC8366] defines the voucher artifact, while the Voucher Request artifact was defined in [RFC8995]. This document defines both a constrained voucher and a constrained voucher-request. They are presented in the order "voucher-request", followed by a "voucher" response as this is the order that they occur in the protocol.

The constrained voucher request MUST be signed by the Pledge. It signs using the private key associated with its IDevID X.509 certificate, or if an IDevID is not available, then the private key associated with its manufacturer-installed raw public key (RPK). Section 12 provides additional details on PKIX-less operations.

The constrained voucher MUST be signed by the MASA.

For the constrained voucher request this document defines two distinct methods for the Pledge to identify the Registrar: using either the Registrar's X.509 certificate, or using a raw public key (RPK) of the Registrar.

For the constrained voucher both methods are supported to indicate (pin) a trusted domain identity: using either a pinned domain X.509 certificate, or a pinned raw public key (RPK).

The BRSKI architectures mandates that the MASA be aware of the capabilities of the pledge. This is not a drawback as the pledges are constructed by a manufacturer which also arranges for the MASA to be aware of the inventory of devices.

The MASA therefore knows if the pledge supports PKIX operations, PKIX format certificates, or if the pledge is limited to Raw Public Keys (RPK). Based upon this, the MASA can select which attributes to use in the voucher for certain operations, like the pinning of the Registrar identity. This is described in more detail in Section 9.2.3, Section 8 and Section 8.3 (for RPK specifically).

## 5. Updates to RFC8366 and RFC8995

This section details the ways in which this document updates other RFCs. The terminology for Updates is taken from [I-D.kuehlewind-update-tag].

This document Updates [RFC8366]. It Extends [RFC8366] by creating a new serialization format, and creates a mechanism to pin Raw Public Key (RPK).

This document Updates [RFC8995]. It Amends [RFC8995]

- \* by clarifying how pinning is done,
- \* adopts clearer explanation of the TLS Server Name Indicator (SNI), see Section 6.1 and Section 7.3
- \* clarifies when new trust anchors should be retrieved (Section 6.6.1),

- \* clarified what kinds of Extended Key Usage attributes are appropriate for each certificate (Section 7.3.1)

It Extends [RFC8995] as follows: \* defines the CoAP version of the BRSKI protocol

- \* makes some messages optional if the results can be inferred from other validations (Section 6.6),
- \* provides the option to return trust anchors in a simpler format (Section 6.6.3)
- \* extends the BRSKI-MASA protocol to carry the new voucher-cose+cbor format.

## 6. BRSKI-EST Protocol

This section describes the constrained BRSKI extensions to EST-coaps [I-D.ietf-ace-coap-est] to transport the voucher between Registrar and Pledge (optionally via a Join Proxy) over CoAP. The extensions are targeting low-resource networks with small packets.

The constrained BRSKI-EST protocol described in this section is between the Pledge and the Registrar only.

### 6.1. Registrar and the Server Name Indicator (SNI)

A DTLS connection is established between the Pledge and the Registrar, similar to the TLS connection described in Section 5.1 of [RFC8995]. This may occur via a Join Proxy as described in Section 6.4. Regardless of the Join Proxy mechanism, the DTLS connection should operate identically.

The SNI issue described below affects [RFC8995] as well, and is reported in errata: <https://www.rfc-editor.org/errata/eid6648>

As the Registrar is discovered by IP address, and typically connected via a Join Proxy, the name of the Registrar is not known to the Pledge. The Pledge will not know what the hostname for the Registrar is, so it cannot do RFC6125 DNS-ID validation on the Registrar's certificate. Instead, it must do validation using the RFC8366 voucher.

As the Pledge does not know the name of the Registrar, the Pledge cannot put any reasonable value into the [RFC6066] Server Name Indicator (SNI). Therefore the Pledge SHOULD omit the SNI extension as per Section 9.2 of [RFC8446].

In some cases, particularly while testing BRSKI, a Pledge may be given the hostname of a particular Registrar to connect to directly. Such a bypass of the discovery process may result in the Pledge taking a different code branch to establish a DTLS connection, and may result in the SNI being inserted by a library. The Registrar MUST ignore any SNI seen.

A primary motivation for making the SNI ubiquitous in the public web is because it allows for multi-tenant hosting of HTTPS sites on a single (scarce) IPv4 address. This consideration does not apply to the server function in the Registrar because:

- \* it uses DTLS and CoAP, not HTTPS
- \* it typically uses IPv6, often [RFC4193] Unique Local Address, which are plentiful
- \* the server port number is typically discovered, so multiple tenants can be accommodated via unique port numbers.

As per Section 3.6.1 of [RFC7030], the Registrar certificate MUST have the Extended Key Usage (EKU) id-kp-cmcRA. This certificate is also used as a TLS Server Certificate, so it MUST also have the EKU id-kp-serverAuth.

## 6.2. TLS Client Certificates: IDevID authentication

As described in Section 5.1 of [RFC8995], the Pledge makes a connection to the Registrar using a TLS Client Certificate for authentication.

Subsequently the Pledge will send a Pledge Voucher Request (PVR).

As explained below in Section 8.1, the "x5bag" element may be used in the RVR to communicate identity of the Registrar to MASA. The Pledge SHOULD NOT use the x5bag attribute in this way in the PVR. A Registrar that processes a PVR with an x5bag attribute MUST ignore it, and MUST use only the TLS Client Certificate extension for authentication of the Pledge.

A situation where the Pledge MAY use the x5bag is for communication of certificate chains to the MASA. This would arise in some vendor-specific situations involving outsourcing of MASA functionality, or rekeying of the IDevID certification authority.

### 6.3. Discovery, URIs and Content Formats

To keep the protocol messages small the EST-coaps and constrained-BRSKI URIs are shorter than the respective EST and BRSKI URIs.

The EST-coaps server URIs differ from the EST URIs by replacing the scheme https by coaps and by specifying shorter resource path names. Below are some examples; the first two using a discovered short path name and the last one using the well-known URI of EST which requires no discovery.

```
coaps://server.example.com/est/<short-name>
coaps://server.example.com/e/<short-name>
coaps://server.example.com/.well-known/est/<short-name>
```

Similarly the constrained BRSKI server URIs differ from the BRSKI URIs by replacing the scheme https by coaps and by specifying shorter resource path names. Below are some examples; the first two using a discovered short path name and the last one using the well-known URI prefix which requires no discovery. This is the same `"/.well-known/brski"` prefix as defined in Section 5 of [RFC8995].

```
coaps://server.example.com/brski/<short-name>
coaps://server.example.com/b/<short-name>
coaps://server.example.com/.well-known/brski/<short-name>
```

Figure 5 in Section 3.2.2 of [RFC7030] enumerates the operations supported by EST, for which Table 1 in Section 5.1 of [I-D.ietf-ace-coap-est] enumerates the corresponding EST-coaps short path names. Similarly, Table 1 below provides the mapping from the supported BRSKI extension URI paths to the constrained-BRSKI URI paths.

=====	=====
BRSKI resource   constrained-BRSKI resource	
=====	=====
/requestvoucher   /rv	
-----	-----
/voucher_status   /vs	
-----	-----
/enrollstatus   /es	
-----	-----

Table 1: BRSKI URI paths mapping to  
Constrained BRSKI URI paths

Note that /requestvoucher indicated above occurs between the Pledge and Registrar (in scope of the BRSKI-EST protocol), but it also occurs between Registrar and MASA. However, as described in Section 6, this section and above table addresses only the BRSKI-EST protocol.

Pledges that wish to discover the available BRSKI bootstrap options/formats, or reduce the size of the CoAP headers by eliminating the "/.well-known/brski" path, can do a discovery operation using [RFC6690] Section 4 by sending a discovery query to the Registrar.

For example, if the Registrar supports a short BRSKI URL (/b) and supports the voucher format "application/voucher-cose+cbor" (TBD3), and status reporting in both CBOR and JSON formats:

```
REQ: GET /.well-known/core?rt=brski*
```

```
RES: 2.05 Content
```

```
Content-Format: 40
```

```
Payload:
```

```
</b>;rt=brski,  
</b/rv>;rt=brski.rv;ct=TBD3,  
</b/vs>;rt=brski.vs;ct="50 60",  
</b/es>;rt=brski.es;ct="50 60"
```

The Registrar is under no obligation to provide shorter URLs, and MAY respond to this query with only the "/.well-known/brski/<short-name>" resources for the short names as defined in Table 1.

Registrars that have implemented shorter URLs MUST also respond in equivalent ways to the corresponding "/.well-known/brski/<short-name>" URLs, and MUST NOT distinguish between them. In particular, a Pledge MAY use the longer and shorter URLs in any combination.

When responding to a discovery request for BRSKI resources, the server MAY in addition return the full resource paths and the content types which are supported by these resources as shown in above example. This is useful when multiple content types are specified for a particular resource on the server. The server responds with only the root path for the BRSKI resources (rt=brski, resource /b in above example) and no others in case the client queries for only rt=brski type resources. (So, a query for rt=brski, without the wildcard character.)

Without discovery, a longer well-known URL can only be used, such as:

```
REQ: GET /.well-known/brski/rv
```

while with discovery of shorter URLs, a request such as:

```
REQ: GET /b/rv
```

is possible.

The return of multiple content-types in the "ct" attribute allows the Pledge to choose the most appropriate one. Note that Content-Format TBD3 ("application/voucher-cose+cbor") is defined in this document.

Content-Format TBD3 MUST be supported by the Registrar for the /rv resource. If the "ct" attribute is not indicated for the /rv resource in the link format description, this implies that at least TBD3 is supported.

Note that this specification allows for voucher-cose+cbor format requests and vouchers to be transmitted over HTTPS, as well as for voucher-cms+json and other formats yet to be defined over CoAP. The burden for this flexibility is placed upon the Registrar. A Pledge on constrained hardware is expected to support a single format only.

The Pledge and MASA need to support one or more formats (at least TBD3) for the voucher and for the voucher request. The MASA needs to support all formats that the Pledge supports.

Section 10 details how the Pledge discovers the Registrar and Join Proxy in different deployment scenarios.

#### 6.3.1. RFC8995 Telemetry Returns

[RFC8995] defines two telemetry returns from the Pledge which are sent to the Registrar. These are the BRSKI Status Telemetry [RFC8995], Section 5.7 and the Enrollment Status Telemetry [RFC8995], Section 5.9.4. These are two POST operations made the by Pledge at two key steps in the process.

[RFC8995] defines the content of these POST operations in CDDL, which are serialized as JSON. This document extends the list of acceptable formats to CBOR as well as JSON, using the rules from [RFC8610].

The existing JSON format is described as CoAP Content-Format 50 ("application/json"), and it MAY be supported. The new CBOR format described as CoAP Content-Format 60 ("application/cbor"), MUST be supported by the Registrar for both the /vs and /es resources.

#### 6.4. Join Proxy options

[I-D.ietf-anima-constrained-join-proxy] specifies a constrained Join Proxy that is optionally placed between Pledge and Registrar. This includes methods for discovery of the Join Proxy by the Pledge and discovery of the Registrar by the Join Proxy.

#### 6.5. Extensions to BRSKI

##### 6.5.1. Discovery

The Pledge discovers an IP address and port number that connects to the Registrar (possibly via a Join Proxy), and it establishes a DTLS connection.

No further discovery of hosts or port numbers is required, but a pledge that can do more than one kind of enrollment (future work offers protocols other than [I-D.ietf-ace-coap-est]), then a pledge may need to use CoAP Discovery to determine what other protocols are available.

A Pledge that only supports the EST-coaps enrollment method SHOULD NOT use discovery for BRSKI resources. It is more efficient to just try the supported enrollment method via the well-known BRSKI/EST-coaps resources. This also avoids the Pledge doing any CoRE Link Format parsing, which is specified in [I-D.ietf-ace-coap-est], Section 4.1.

The Registrar MUST support all of the EST resources at their default ".well-known" locations (on the specified port) as well as any server-specific shorter form that might also be supported.

However, when discovery is being done by the Pledge, it is possible for the Registrar to return references to resources which are on different port numbers. The Registrar SHOULD NOT use different ports numbers by default, because a Pledge that is connected via a Join Proxy can only access a single UDP port. A Registrar configured to never use Join Proxies MAY be configured to use multiple port numbers. Therefore a Registrar MUST host all discoverable BRSKI resources on the same (UDP) server port that the Pledge's DTLS connection is using. Using the same UDP server port for all resources allows the Pledge to continue via the same DTLS connection which is more efficient.

##### 6.5.2. CoAP responses

[RFC8995], Section 5 defines a number of HTTP response codes that the Registrar is to return when certain conditions occur.

The 401, 403, 404, 406 and 415 response codes map directly to CoAP codes 4.01, 4.03, 4.04, 4.06 and 4.15.

The 202 Retry process which occurs in the voucher request, is to be handled in the same way as Section 5.7 of [I-D.ietf-ace-coap-est] process for Delayed Responses.

## 6.6. Extensions to EST-coaps

This document extends [I-D.ietf-ace-coap-est], and it inherits the functions described in that document: specifically, the mandatory Simple (Re-)Enrollment (/sen and /sren) and Certification Authority certificates request (/crtcs). Support for CSR Attributes Request (/att) and server-side key generation (/skg, /skc) remains optional for the EST server.

Collecting the resource definitions from both [RFC8995], [RFC7030], and [I-D.ietf-ace-coap-est] results in the following shorter forms of URI paths for the commonly used resources:

=====	=====	=====
BRSKI + EST	Constrained-BRSKI + EST	Well-known URI namespace
=====	=====	=====
/requestvoucher	/rv	brski
/voucher_status	/vs	brski
/csrattrs	/att	est
/simpleenroll	/sen	est
/cacerts	/crtcs	est
/enrollstatus	/es	brski
/simplereenroll	/sren	est
=====	=====	=====

Table 2: BRSKI/EST URI paths mapping to Constrained BRSKI/EST short URI paths

### 6.6.1. Pledge Extensions

This section defines extensions to the BRSKI Pledge, which are applicable during the BRSKI bootstrap procedure. A Pledge which only supports the EST-coaps enrollment method, SHOULD NOT use discovery for EST-coaps resources, because it is more efficient to enroll (e.g. /sen) via the well-known EST resource on the current DTLS connection. This avoids an additional round-trip of packets and avoids the Pledge having to unnecessarily implement CoRE Link Format parsing.

A constrained Pledge SHOULD NOT perform the optional EST "CSR attributes request" (/att) to minimize network traffic. The Pledge selects which attributes to include in the CSR.

One or more Subject Distinguished Name fields MUST be included. If the Pledge has no specific information on what attributes/fields are desired in the CSR, it MUST use the Subject Distinguished Name fields from its LDevID unmodified. The Pledge can receive such information via the voucher (encoded in a vendor-specific way) or via some other, out-of-band means.

A constrained Pledge MAY use the following optimized EST-coaps procedure to minimize network traffic.

1. if the voucher, that validates the current Registrar, contains a single pinned domain CA certificate, the Pledge provisionally considers this certificate as the EST trust anchor, as if it were the result of "CA certificates request" (/crts) to the Registrar.
2. Using this CA certificate as trust anchor it proceeds with EST simple enrollment (/sen) to obtain its provisionally trusted LDevID certificate.
3. If the Pledge validates that the trust anchor CA was used to sign its LDevID certificate, the Pledge accepts the pinned domain CA certificate as the legitimate trust anchor CA for the Registrar's domain and accepts the associated LDevID certificate.
4. If the trust anchor CA was NOT used to sign its LDevID certificate, the Pledge MUST perform an actual "CA certificates request" (/crts) to the EST server to obtain the EST CA trust anchor(s) since these can differ from the (temporary) pinned domain CA.

5. When doing this /crtts request, the Pledge MAY use a CoAP Accept Option with value TBD287 ("application/pkix-cert") to limit the number of returned EST CA trust anchors to only one. A constrained Pledge MAY support only this format in a /crtts response, per Section 5.3 of [I-D.ietf-ace-coap-est].
6. If the Pledge cannot obtain the single CA certificate or the finally validated CA certificate cannot be chained to the LDevID certificate, then the Pledge MUST abort the enrollment process and report the error using the enrollment status telemetry (/es).

Note that even though the Pledge may avoid performing any /crtts request using the above EST-coaps procedure during bootstrap, it SHOULD support retrieval of the trust anchor CA periodically as detailed in the next section.

#### 6.6.2. EST-client Extensions

This section defines extensions to EST-coaps clients, used after the BRSKI bootstrap procedure is completed. (Note that such client is not called "Pledge" in this section, since it is already enrolled into the domain.) A constrained EST-coaps client MAY support only the Content-Format TBD287 ("application/pkix-cert") in a /crtts response, per Section 5.3 of [I-D.ietf-ace-coap-est]. In this case, it can only store one trust anchor of the domain.

An EST-coaps client that has an idea of the current time (internally, or via NTP) SHOULD consider the validity time of the trust anchor CA, and MAY begin requesting a new trust anchor CA using the /crtts request when the CA has 50% of its validity time (notAfter - notBefore) left. A client without access to the current time cannot decide if the trust anchor CA has expired, and SHOULD poll periodically for a new trust anchor using the /crtts request at an interval of approximately 1 month. An EST-coaps server SHOULD include the CoAP ETag Option in every response to a /crtts request, to enable clients to perform low-overhead validation whether their trust anchor CA is still valid. The EST-coaps client SHOULD store the ETag resulting from a /crtts response in memory and SHOULD use this value in an ETag Option in its next GET /crtts request.

The above-mentioned limitation that an EST-coaps client may support only one trust anchor CA is not an issue in case the domain trust anchor remains stable. However, special consideration is needed for cases where the domain trust anchor can change over time. Such a change may happen due to relocation of the client device to a new domain, or due to key update of the trust anchor as described in [RFC4210], Section 4.4.

From the client's viewpoint, a trust anchor change typically happens during EST re-enrollment: a change of domain CA requires all devices operating under the old domain CA to acquire a new LDevID issued by the new domain CA. A client's re-enrollment may be triggered by various events, such as an instruction to re-enroll sent by a domain entity, or an imminent expiry of its LDevID certificate. How the re-enrollment is explicitly triggered on the client by a domain entity, such as a commissioner or a Registrar, is out of scope of this specification.

The mechanism described in [RFC4210], Section 4.4 for Root CA key update requires four certificates: OldWithOld, OldWithNew, NewWithOld, and NewWithNew. The OldWithOld certificate is already stored in the EST client's trust store. The NewWithNew certificate will be distributed as the single certificate in a /crt response, during EST re-enrollment. Since the EST client can only accept a single certificate in a /crt response it implies that the EST client cannot obtain the certificates OldWithNew and NewWithOld in this way, to perform the complete verification of the new domain CA. Instead, the client only verifies the EST server (Registrar) using its old domain CA certificate in its trust store as detailed below, and based on this trust in the active and valid DTLS connection it automatically trusts the new (NewWithNew) domain CA certificate that the EST server provides in the /crt response.

In this manner, even during rollover of trust anchors, it is possible to have only a single trust anchor provided in a /crt response.

During the period of the certificate renewal, it is not possible to create new communication channels between devices with NewCA certificates devices with OldCA certificates. One option is that devices should avoid restarting existing DTLS or OSCORE connections during this interval that new certificates are being deployed. The recommended period for certificate renewal is 24 hours. For re-enrollment, the constrained EST-coaps client MUST support the following EST-coaps procedure, where optional re-enrollment to a new domain is under control of the Registrar:

1. The client connects with DTLS to the Registrar, and authenticates with its present domain certificate (LDevID certificate) as usual. The Registrar authenticates itself with its domain certificate that is trusted by the client, i.e. it chains to the single trust anchor that the client has stored. This is the "old" trust anchor, the one that will be eventually replaced in case the Registrar decides to re-enroll the client into a new domain.

2. The client performs the simple re-enrollment request (/sren) and upon success it obtains a new LDevID.
3. The client verifies the new LDevID against its (single) existing domain trust anchor. If it chains successfully, this means the trust anchor did not change and the client MAY skip retrieving the current CA certificate using the "CA certificates request" (/crts). If it does not chain successfully, this means the trust anchor was changed/updated and the client then MUST retrieve the new domain trust anchor using the "CA certificates request" (/crts).
4. If the client retrieved a new trust anchor in step 3, then it MUST verify that the new trust anchor chains with the new LDevID certificate it obtained in step 2. If it chains successfully, the client stores both, accepts the new LDevID certificate and stops using its prior LDevID certificate. If it does not chain successfully, the client MUST NOT update its LDevID certificate, it MUST NOT update its (single) domain trust anchor, and the client MUST abort the enrollment process and report the error to the Registrar using enrollment status telemetry (/es).

Note that even though the EST-coaps client may skip the /crts request in step 3, it SHOULD support retrieval of the trust anchor CA periodically as detailed earlier in this section.

#### 6.6.3. Registrar Extensions

A Registrar SHOULD host any discoverable EST-coaps resources on the same (UDP) server port that the Pledge's DTLS initial connection is using. This avoids the overhead of the Pledge reconnecting using DTLS, when it performs EST enrollment after the BRSKI voucher request.

The Content-Format 50 (application/json) MUST be supported and 60 (application/cbor) MUST be supported by the Registrar for the /vs and /es resources.

Content-Format TBD3 MUST be supported by the Registrar for the /rv resource.

When a Registrar receives a "CA certificates request" (/crts) request with a CoAP Accept Option with value TBD287 ("application/pkix-cert") it SHOULD return only the single CA certificate that is the envisioned or actual authority for the current, authenticated Pledge making the request.

If the Pledge included in its request an Accept Option for only the TBD287 ("application/pkix-cert") Content Format, but the domain has been configured to operate with multiple CA trust anchors only, then the Registrar returns a 4.06 Not Acceptable error to signal that the Pledge needs to use the Content Format 281 ("application/pkcs7-mime; smime-type=certs-only") to retrieve all the certificates.

If the current authenticated client is an EST-coaps client that was already enrolled in the domain, and the Registrar is configured to assign this client to a new domain CA trust anchor during the next EST re-enrollment procedure, then the Registrar MUST respond with the new domain CA certificate in case the client performs the "CA Certificates request" (/crt) with an Accept Option for TBD287 only. This signals the client that a new domain is assigned to it. The client follows the procedure as defined in Section 6.6.2.

#### 6.7. DTLS handshake fragmentation Considerations

DTLS includes a mechanism to fragment the handshake messages. This is described in Section 4.4 of [I-D.ietf-tls-dtls13]. The protocol described in this document will often be used with a Join Proxy described in [I-D.ietf-anima-constrained-join-proxy]. The Join Proxy will need some overhead, while the maximum packet sized guaranteed on 802.15.4 networks is 1280 bytes. It is RECOMMENDED that a PMTU of 1024 bytes be assumed for the DTLS handshake. It is unlikely that any Packet Too Big indications [RFC4443] will be relayed by the Join Proxy.

During the operation of the constrained BRSKI-EST protocol, the CoAP Blockwise transfer mechanism will be used when message sizes exceed the PMTU. A Pledge/EST-client on a constrained network MUST use the (D)TLS maximum fragment length extension ("max\_fragment\_length") defined in Section 4 of [RFC6066] with the maximum fragment length set to a value of either  $2^9$  or  $2^{10}$ .

### 7. BRSKI-MASA Protocol

This section describes extensions to and clarifications of the BRSKI-MASA protocol between Registrar and MASA.

#### 7.1. Protocol and Formats

Section 5.4 of [RFC8995] describes a connection between the Registrar and the MASA as being a normal TLS connection using HTTPS. This document does not change that. The Registrar MUST use the format "application/voucher-cose+cbor" in its voucher request to MASA, when the Pledge uses this format in its requests to the Registrar [RFC8995].

The MASA only needs to support formats for which there are Pledges that use that format.

The Registrar MUST use the same format for the RVR as the Pledge used for its PVR.

The Registrar indicates the voucher format it wants to receive from MASA using the HTTP Accept header. This format MUST be the same as the format of the PVR, so that the Pledge can parse it.

At the moment of writing the creation of coaps based MASAs is deemed unrealistic. The use of CoAP for the BRSKI-MASA connection can be the subject of another document. Some consideration was made to specify CoAP support for consistency, but:

- \* the Registrar is not expected to be so constrained that it cannot support HTTPS client connections.
- \* the technology and experience to build Internet-scale HTTPS responders (which the MASA is) is common, while the experience doing the same for CoAP is much less common.
- \* a Registrar is likely to provide onboarding services to both constrained and non-constrained devices. Such a Registrar would need to speak HTTPS anyway.
- \* a manufacturer is likely to offer both constrained and non-constrained devices, so there may in practice be no situation in which the MASA could be CoAP-only. Additionally, as the MASA is intended to be a function that can easily be outsourced to a third-party service provider, reducing the complexity would also seem to reduce the cost of that function.
- \* security-related considerations: see Section 14.6.

## 7.2. Registrar Voucher Request

If the PVR contains a proximity assertion, the Registrar MUST propagate this assertion into the RVR by including the "assertion" field with the value "proximity". This conforms to the example in Section 3.3 of [RFC8995] of carrying the assertion forward.

## 7.3. MASA and the Server Name Indicator (SNI)

A TLS/HTTPS connection is established between the Registrar and MASA.

Section 5.4 of [RFC8995] explains this process, and there are no externally visible changes. A MASA that supports the unconstrained voucher formats should be able to support constrained voucher formats equally well.

There is no requirement that a single MASA be used for both constrained and unconstrained voucher requests: the choice of MASA is determined by the id-mod-MASAURLExtn2016 extension contained in the IDDevID.

The Registrar MUST do [RFC6125] DNS-ID checks on the contents of the certificate provided by the MASA.

In contrast to the Pledge/Registrar situation, the Registrar always knows the name of the MASA, and MUST always include an [RFC6066] Server Name Indicator. The SNI is optional in TLS1.2, but common. The SNI is considered mandatory with TLS1.3.

The presence of the SNI is needed by the MASA, in order for the MASA's server to host multiple tenants (for different customers).

The Registrar SHOULD use a TLS Client Certificate to authenticate to the MASA per Section 5.4.1 of [RFC8995]. If the certificate that the Registrar uses is marked as a id-kp-cmcRA certificate, via Extended Key Usage, then it MUST also have the id-kp-clientAuth EKU attribute set.

#### 7.3.1. Registrar Certificate Requirement

In summary for typical Registrar use, where a single Registrar certificate is used, then the certificate MUST have EKU of: id-kp-cmcRA, id-kp-serverAuth, id-kp-clientAuth.

### 8. Pinning in Voucher Artifacts

The voucher is a statement by the MASA for use by the Pledge that provides the identity of the Pledge's owner. This section describes how the owner's identity is determined and how it is specified within the voucher.

#### 8.1. Registrar Identity Selection and Encoding

Section 5.5 of [RFC8995] describes BRSKI policies for selection of the owner identity. It indicates some of the flexibility that is possible for the Registrar, and recommends the Registrar to include only certificates in the voucher request (CMS) signing structure that participate in the certificate chain that is to be pinned.

The MASA is expected to evaluate the certificates included by the Registrar in its voucher request, forming them into a chain with the Registrar's (signing) identity on one end. Then, it pins a certificate selected from the chain. For instance, for a domain with a two-level certification authority (see Figure 1), where the voucher-request has been signed by "Registrar", its signing structure includes two additional CA certificates. The arrows in the figure indicate the issuing of a certificate, i.e. author of (1) issued (2) and author of (2) issued (3).

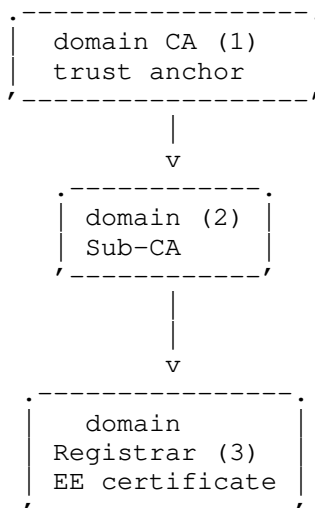


Figure 1: Two-Level CA PKI

When the Registrar is using a COSE-signed constrained voucher request towards MASA, instead of a regular CMS-signed voucher request, the COSE\_Sign1 object contains a protected and an unprotected header. The Registrar MUST place all the certificates needed to validate the signature chain from the Registrar on the RVR in an "x5bag" attribute in the unprotected header [I-D.ietf-cose-x509].

The "x5bag" attribute is very important as it provides the required signals from the Registrar to control what identity is pinned in the resulting voucher. This is explained in the next section.

## 8.2. MASA Pinning Policy

The MASA, having assembled and verified the chain in the signing structure of the voucher request needs to select a certificate to pin. (For the case that only the Registrar's End-Entity certificate is included, only this certificate can be selected and this section does not apply.) The BRSKI policy for pinning by the MASA as described in Section 5.5.2 of [RFC8995] leaves much flexibility to the manufacturer.

The present document adds the following rules to the MASA pinning policy to reduce the network load:

1. for a voucher containing a nonce, it SHOULD select the most specific (lowest-level) CA certificate in the chain.
2. for a nonceless voucher, it SHOULD select the least-specific (highest-level) CA certificate in the chain that is allowed under the MASA's policy for this specific domain.

The rationale for 1. is that in case of a voucher with nonce, the voucher is valid only in scope of the present DTLS connection between Pledge and Registrar anyway, so there is no benefit to pin a higher-level CA. By pinning the most specific CA the constrained Pledge can validate its DTLS connection using less crypto operations. The rationale for pinning a CA instead of the Registrar's End-Entity certificate directly is based on the following benefit on constrained networks: the pinned certificate in the voucher can in common cases be re-used as a Domain CA trust anchor during the EST enrollment and during the operational phase that follows after EST enrollment, as explained in Section 6.6.1.

The rationale for 2. follows from the flexible BRSKI trust model for, and purpose of, nonceless vouchers (Sections 5.5.\* and 7.4.1 of [RFC8995]).

Referring to Figure 1 of a domain with a two-level certification authority, the most specific CA ("Sub-CA") is the identity that is pinned by MASA in a nonced voucher. A Registrar that wished to have only the Registrar's End-Entity certificate pinned would omit the "domain CA" and "Sub-CA" certificates from the voucher-request.

In case of a nonceless voucher, depending on the trust level, the MASA pins the "Registrar" certificate (low trust in customer), or the "Sub-CA" certificate (in case of medium trust, implying that any Registrar of that sub-domain is acceptable), or even the "domain CA" certificate (in case of high trust in the customer, and possibly a pre-agreed need of the customer to obtain flexible long-lived vouchers).

### 8.3. Pinning of Raw Public Keys

Specifically for constrained use cases, the pinning of the raw public key (RPK) of the Registrar is also supported in the constrained voucher, instead of an X.509 certificate. If an RPK is pinned it MUST be the RPK of the Registrar.

When the Pledge is known by MASA to support RPK but not X.509 certificates, the voucher produced by the MASA pins the RPK of the Registrar in either the "pinned-domain-pubk" or "pinned-domain-pubk-sha256" field of a voucher. This is described in more detail in Section 9.2.3. A Pledge that does not support X.509 certificates cannot use EST to enroll; it has to use another method for enrollment without certificates and the Registrar has to support this method also. It is possible that the Pledge will not enroll, but instead only a network join operation will occur (See [RFC9031]). How the Pledge discovers this method and details of the enrollment method are out of scope of this document.

When the Pledge is known by MASA to support PKIX format certificates, the "pinned-domain-cert" field present in a voucher typically pins a domain certificate. That can be either the End-Entity certificate of the Registrar, or the certificate of a domain CA of the Registrar's domain as specified in Section 8.2. However, if the Pledge is known to also support RPK pinning and the MASA intends to identify the Registrar in the voucher (not the CA), then MASA MUST pin the RPK (RPK3 in Figure 2) of the Registrar instead of the Registrar's End-Entity certificate to save space in the voucher.

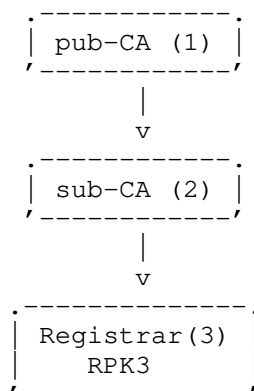


Figure 2: Raw Public Key pinning

#### 8.4. Considerations for use of IDevID-Issuer

[RFC8366] and [RFC8995] defines the idevid-issuer attribute for voucher and voucher-request (respectively), but they summarily explain when to use it.

The use of idevid-issuer is provided so that the serial-number to which the issued voucher pertains can be relative to the entity that issued the devices' IDevID. In most cases there is a one to one relationship between the trust anchor that signs vouchers (and is trusted by the pledge), and the Certification Authority that signs the IDevID. In that case, the serial-number in the voucher must refer to the same device as the serial-number that is in IDevID certificate.

However, there situations where the one to one relationship may be broken. This occurs whenever a manufacturer has a common MASA, but different products (on different assembly lines) are produced with identical serial numbers. A system of serial numbers which is just a simple counter is a good example of this. A system of serial numbers where there is some prefix relating the product type does not fit into this, even if the lower digits are a counter.

It is not possible for the Pledge or the Registrar to know which situation applies. The question to be answered is whether or not to include the idevid-issuer in the PVR and the RVR. A second question arises as to what the format of the idevid-issuer contents are.

Analysis of the situation shows that the pledge never needs to include the idevid-issuer in its PVR, because the pledge's IDevID certificate is available to the Registrar, and the Authority Key Identifier is contained within that. The pledge therefore has no need to repeat this.

For the RVR, the Registrar has to examine the pledge's IDevID certificate to discover the serial number for the Registrar's Voucher Request (RVR). This is clear in Section 5.5 of [RFC8995]. That section also clarifies that the idevid-issuer is to be included in the RVR.

Concerning the Authority Key Identifier, [RFC8366] specifies that the entire object i.e. the extnValue OCTET STRING is to be included: comprising the AuthorityKeyIdentifier, SEQUENCE, Choice as well as the OCTET STRING that is the keyIdentifier.

## 9. Artifacts

This section describes for both the voucher request and the voucher first the abstract (tree) definition as explained in [RFC8340]. This provides a high-level view of the contents of each artifact.

Then the assigned SID values are presented. These have been assigned using the rules in [I-D.ietf-core-sid].

### 9.1. Voucher Request artifact

#### 9.1.1. Tree Diagram

The following diagram is largely a duplicate of the contents of [RFC8366], with the addition of the fields proximity-registrar-pubk, proximity-registrar-pubk-sha256, proximity-registrar-cert, and prior-signed-voucher-request.

prior-signed-voucher-request is only used between the Registrar and the MASA. proximity-registrar-pubk or proximity-registrar-pubk-sha256 optionally replaces proximity-registrar-cert for the most constrained cases where RPK is used by the Pledge.

```
module: ietf-voucher-request-constrained
```

```
  grouping voucher-request-constrained-grouping
```

```
    +-- voucher
```

```
      +-- created-on?                yang:date-and-time
      +-- expires-on?               yang:date-and-time
      +-- assertion                  enumeration
      +-- serial-number              string
      +-- idevid-issuer?             binary
      +-- pinned-domain-cert?        binary
      +-- domain-cert-revocation-checks? boolean
      +-- nonce?                    binary
      +-- last-renewal-date?         yang:date-and-time
      +-- proximity-registrar-pubk?  binary
      +-- proximity-registrar-pubk-sha256? binary
      +-- proximity-registrar-cert?  binary
      +-- prior-signed-voucher-request? binary
```

#### 9.1.2. SID values

```
  SID Assigned to
```

```
-----
2501 data /ietf-voucher-request-constrained:voucher
2502 data .../assertion
2503 data .../created-on
2504 data .../domain-cert-revocation-checks
2505 data .../expires-on
2506 data .../idevid-issuer
2507 data .../last-renewal-date
2508 data /ietf-voucher-request-constrained:voucher/nonce
2509 data .../pinned-domain-cert
2510 data .../prior-signed-voucher-request
2511 data .../proximity-registrar-cert
2513 data .../proximity-registrar-pubk
2512 data .../proximity-registrar-pubk-sha256
2514 data .../serial-number
```

WARNING, obsolete definitions

The "assertion" attribute is an enumerated type [RFC8366], and the current PYANG tooling does not document the valid values for this attribute. In the JSON serialization, the literal strings from the enumerated types are used so there is no ambiguity. In the CBOR serialization, a small integer is used. This following values are documented here, but the YANG module should be considered authoritative. No IANA registry is provided or necessary because the YANG module provides for extensions.

Integer	Assertion Type
0	verified
1	logged
2	proximity

Table 3: CBOR integers  
for the "assertion"  
attribute enum

### 9.1.3. YANG Module

In the voucher-request-constrained YANG module, the voucher is "augmented" within the "used" grouping statement such that one continuous set of SID values is generated for the voucher-request-constrained module name, all voucher attributes, and the voucher-request-constrained attributes. Two attributes of the voucher are "refined" to be optional.

```
<CODE BEGINS> file "ietf-voucher-request-constrained@2021-04-15.yang"
module ietf-voucher-request-constrained {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-voucher-request-constrained";
  prefix "constrained";

  import ietf-restconf {
    prefix rc;
    description
      "This import statement is only present to access
       the yang-data extension defined in RFC 8040.";
    reference "RFC 8040: RESTCONF Protocol";
  }

  import ietf-voucher {
    prefix "v";
  }

  organization
    "IETF ANIMA Working Group";

  contact
    "WG Web:  <http://tools.ietf.org/wg/anima/>
```

WG List: <mailto:anima@ietf.org>  
Author: Michael Richardson  
<mailto:mcr+ietf@sandelman.ca>  
Author: Peter van der Stok  
<mailto:consultancy@vanderstok.org>  
Author: Panos Kampanakis  
<mailto:pkampana@cisco.com>;

#### description

"This module defines the format for a voucher request, which is produced by a pledge to request a voucher. The voucher-request is sent to the potential owner's Registrar, which in turn sends the voucher request to the manufacturer or its delegate (MASA).

A voucher is then returned to the pledge, binding the pledge to the owner. This is a constrained version of the voucher-request present in  
{[I-D.ietf-anima-bootstrap-keyinfra]}

This version provides a very restricted subset appropriate for very constrained devices. In particular, it assumes that nonce-ful operation is always required, that expiration dates are rather weak, as no clocks can be assumed, and that the Registrar is identified by either a pinned Raw Public Key of the Registrar, or by a pinned X.509 certificate of the Registrar or domain CA.

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in the module text are to be interpreted as described in RFC 2119."

```
revision "2021-04-15" {  
  description  
    "Initial version";  
  reference  
    "RFC XXXX: Voucher Profile for Constrained Devices";  
}
```

```
rc:yang-data voucher-request-constrained-artifact {  
  // YANG data template for a voucher.  
  uses voucher-request-constrained-grouping;  
}
```

```
// Grouping defined for future usage  
grouping voucher-request-constrained-grouping {  
  description
```

```
"Grouping to allow reuse/extensions in future work.";

uses v:voucher-artifact-grouping {

  refine voucher/created-on {
    mandatory false;
  }

  refine voucher/pinned-domain-cert {
    mandatory false;
  }

  augment "voucher" {
    description "Base the constrained voucher-request upon the
      regular one";

    leaf proximity-registrar-pubk {
      type binary;
      description
        "The proximity-registrar-pubk replaces
        the proximity-registrar-cert in constrained uses of
        the voucher-request.
        The proximity-registrar-pubk is the
        Raw Public Key of the Registrar. This field is encoded
        as specified in RFC7250, section 3.
        The ECDSA algorithm MUST be supported.
        The EdDSA algorithm as specified in
        draft-ietf-tls-rfc4492bis-17 SHOULD be supported.
        Support for the DSA algorithm is not recommended.
        Support for the RSA algorithm is a MAY, but due to
        size is discouraged.";
    }

    leaf proximity-registrar-pubk-sha256 {
      type binary;
      description
        "The proximity-registrar-pubk-sha256
        is an alternative to both
        proximity-registrar-pubk and pinned-domain-cert.
        In many cases the public key of the domain has already
        been transmitted during the key agreement protocol,
        and it is wasteful to transmit the public key another
        two times.
        The use of a hash of public key info, at 32-bytes for
        sha256 is a significant savings compared to an RSA
        public key, but is only a minor savings compared to
        a 256-bit ECDSA public-key."
    }
  }
}
```

```
        Algorithm agility is provided by extensions to this
        specification which may define a new leaf for another
        hash type.";
    }

    leaf proximity-registrar-cert {
        type binary;
        description
            "An X.509 v3 certificate structure as specified by
            RFC 5280,
            Section 4 encoded using the ASN.1 distinguished encoding
            rules (DER), as specified in ITU-T X.690.

            The first certificate in the Registrar TLS server
            certificate_list sequence (see [RFC5246]) presented by
            the Registrar to the Pledge. This field or one of its
            alternatives MUST be populated in a
            Pledge's voucher request if the proximity assertion is
            populated.";
    }

    leaf prior-signed-voucher-request {
        type binary;
        description
            "If it is necessary to change a voucher, or re-sign and
            forward a voucher that was previously provided along a
            protocol path, then the previously signed voucher
            SHOULD be included in this field.

            For example, a pledge might sign a proximity voucher,
            which an intermediate registrar then re-signs to
            make its own proximity assertion. This is a simple
            mechanism for a chain of trusted parties to change a
            voucher, while maintaining the prior signature
            information.

            The pledge MUST ignore all prior voucher information
            when accepting a voucher for imprinting. Other
            parties MAY examine the prior signed voucher
            information for the purposes of policy decisions.
            For example, this information could be useful to a
            MASA to determine that both pledge and registrar
            agree on proximity assertions. The MASA SHOULD
            remove all prior-signed-voucher-request information when
            signing a voucher for imprinting so as to minimize the
            final voucher size.";
    }
}
```

```

    }
  }
}
<CODE ENDS>

```

#### 9.1.4. Example voucher request artifact

Below is a CBOR serialization of an example constrained voucher request from a Pledge to a Registrar, shown in CBOR diagnostic notation. The enum value of the assertion field is calculated to be 2 by following the algorithm described in section 9.6.4.2 of [RFC7950].

```

{
  2501: {
    +2 : "2016-10-07T19:31:42Z", / SID=2503, created-on /
    +4 : "2016-10-21T19:31:42Z", / SID=2505, expires-on /
    +1 : 2, / SID=2502, assertion "proximity" /
    +13: "JADA123456789", / SID=2514, serial-number /
    +5 : h'08C2BF36....B3D2B3', / SID=2506, idevid-issuer /
    +10: h'30820275....82c35f', / SID=2511, proximity-registrar-cert/
    +3 : true, / SID=2504, domain-cert
    +6 : "2017-10-07T19:31:42Z" / SID=2507, last-renewal-date /
  }
}

```

#### 9.2. Voucher artifact

The voucher's primary purpose is to securely assign a Pledge to an owner. The voucher informs the Pledge which entity it should consider to be its owner.

##### 9.2.1. Tree Diagram

The following diagram is largely a duplicate of the contents of [RFC8366], with only the addition of the fields pinned-domain-pubk and pinned-domain-pubk-sha256.

module: ietf-voucher-constrained

grouping voucher-constrained-grouping

```

+-- voucher
  +-- created-on?          yang:date-and-time
  +-- expires-on?         yang:date-and-time
  +-- assertion            enumeration
  +-- serial-number        string
  +-- idevid-issuer?       binary
  +-- pinned-domain-cert?  binary
  +-- domain-cert-revocation-checks? boolean
  +-- nonce?              binary
  +-- last-renewal-date?   yang:date-and-time
  +-- pinned-domain-pubk?  binary
  +-- pinned-domain-pubk-sha256? binary

```

### 9.2.2. SID values

SID Assigned to

```

-----
2451 data /ietf-voucher-constrained:voucher
2452 data /ietf-voucher-constrained:voucher/assertion
2453 data /ietf-voucher-constrained:voucher/created-on
2454 data .../domain-cert-revocation-checks
2455 data /ietf-voucher-constrained:voucher/expires-on
2456 data /ietf-voucher-constrained:voucher/idevid-issuer
2457 data .../last-renewal-date
2458 data /ietf-voucher-constrained:voucher/nonce
2459 data .../pinned-domain-cert
2460 data .../pinned-domain-pubk
2461 data .../pinned-domain-pubk-sha256
2462 data /ietf-voucher-constrained:voucher/serial-number

```

WARNING, obsolete definitions

The "assertion" enumerated attribute is numbered as per Section 9.1.2.

### 9.2.3. YANG Module

In the voucher-constrained YANG module, the voucher is "augmented" within the "used" grouping statement such that one continuous set of SID values is generated for the voucher-constrained module name, all voucher attributes, and the voucher-constrained attributes. Two attributes of the voucher are "refined" to be optional.

```
<CODE BEGINS> file "ietf-voucher-constrained@2021-04-15.yang"
module ietf-voucher-constrained {
  yang-version 1.1;

  namespace
    "urn:ietf:params:xml:ns:yang:ietf-voucher-constrained";
  prefix "constrained";

  import ietf-restconf {
    prefix rc;
    description
      "This import statement is only present to access
       the yang-data extension defined in RFC 8040.";
    reference "RFC 8040: RESTCONF Protocol";
  }

  import ietf-voucher {
    prefix "v";
  }

  organization
    "IETF ANIMA Working Group";

  contact
    "WG Web:    <http://tools.ietf.org/wg/anima/>
    WG List:    <mailto:anima@ietf.org>
    Author:     Michael Richardson
                <mailto:mcr+ietf@sandelman.ca>
    Author:     Peter van der Stok
                <mailto:consultancy@vanderstok.org>
    Author:     Panos Kampanakis
                <mailto:pkampana@cisco.com>";

  description
    "This module defines the format for a voucher, which
     is produced by a pledge's manufacturer or its delegate
     (MASA) to securely assign one or more pledges to an 'owner',
     so that a pledge may establish a secure connection to the
     owner's network infrastructure.

     This version provides a very restricted subset appropriate
     for very constrained devices.
     In particular, it assumes that nonce-ful operation is
     always required, that expiration dates are rather weak, as no
     clocks can be assumed, and that the Registrar is identified
     by either a pinned Raw Public Key of the Registrar, or by a
     pinned X.509 certificate of the Registrar or domain CA."
```

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'MAY', and 'OPTIONAL' in the module text are to be interpreted as described in RFC 2119.";

```
revision "2021-04-15" {
  description
    "Initial version";
  reference
    "RFC XXXX: Voucher Profile for Constrained Devices";
}

rc:yang-data voucher-constrained-artifact {
  // YANG data template for a voucher.
  uses voucher-constrained-grouping;
}

// Grouping defined for future usage
grouping voucher-constrained-grouping {
  description
    "Grouping to allow reuse/extensions in future work.";

  uses v:voucher-artifact-grouping {

    refine voucher/created-on {
      mandatory false;
    }

    refine voucher/pinned-domain-cert {
      mandatory false;
    }

    augment "voucher" {
      description "Base the constrained voucher
                  upon the regular one";
      leaf pinned-domain-pubk {
        type binary;
        description
          "The pinned-domain-pubk may replace the
           pinned-domain-cert in constrained uses of
           the voucher. The pinned-domain-pubk
           is the Raw Public Key of the Registrar.
           This field is encoded as a Subject Public Key Info block
           as specified in RFC7250, in section 3.
           The ECDSA algorithm MUST be supported.
           The EdDSA algorithm as specified in
           draft-ietf-tls-rfc4492bis-17 SHOULD be supported.
           Support for the DSA algorithm is not recommended."
```

```

    Support for the RSA algorithm is a MAY.";
}

leaf pinned-domain-pubk-sha256 {
    type binary;
    description
        "The pinned-domain-pubk-sha256 is a second
        alternative to pinned-domain-cert. In many cases the
        public key of the domain has already been transmitted
        during the key agreement process, and it is wasteful
        to transmit the public key another two times.
        The use of a hash of public key info, at 32-bytes for
        sha256 is a significant savings compared to an RSA
        public key, but is only a minor savings compared to
        a 256-bit ECDSA public-key.
        Algorithm agility is provided by extensions to this
        specification which can define a new leaf for another
        hash type.";
}
}
}
}
}
<CODE ENDS>

```

#### 9.2.4. Example voucher artifacts

Below the CBOR serialization of an example constrained voucher is shown in CBOR diagnostic notation. The enum value of the assertion field is calculated to be zero by following the algorithm described in section 9.6.4.2 of [RFC7950].

```

2451: {
+2 : "2016-10-07T19:31:42Z", / SID = 2453, created-on /
+4 : "2016-10-21T19:31:42Z", / SID = 2455, expires-on /
+1 : 0, / SID = 2452, assertion "verified" /
+11: "JADA123456789", / SID = 2462, serial-number /
+5 : h'E40393B4....68A3', / SID = 2456, idevid-issuer /
+8 : h'30820275....C35F', / SID = 2459, pinned-domain-cert/
+3 : true, / SID = 2454, domain-cert /
/
-revocation-checks /
+6 : "2017-10-07T19:31:42Z" / SID = 2457, last-renewal-date /
}
}

```

### 9.3. Signing voucher and voucher-request artifacts with COSE

The COSE\_Sign1 structure is discussed in Section 4.2 of [I-D.ietf-cose-rfc8152bis-struct]. The CBOR object that carries the body, the signature, and the information about the body and signature is called the COSE\_Sign1 structure. It is used when only one signature is used on the body.

Support for ECDSA with SHA2-256 using curve secp256r1 (aka prime256k1) is RECOMMENDED. Most current low power hardware has support for acceleration of this algorithm. Future hardware designs could omit this in favour of a newer algorithms. This is the ES256 keytype from Table 1 of [I-D.ietf-cose-rfc8152bis-algs]. Support for curve secp256k1 is OPTIONAL.

Support for EdDSA using Curve 25519 is RECOMMENDED in new designs if hardware support is available. This is keytype EDDSA (-8) from Table 2 of [I-D.ietf-cose-rfc8152bis-algs]. A "crv" parameter is necessary to specify the Curve, which from Table 18. The 'kty' field MUST be present, and it MUST be 'OKP'. (Table 17)

A transition towards EdDSA is occurring in the industry. Some hardware can accelerate only some algorithms with specific curves, other hardware can accelerate any curve, and still other kinds of hardware provide a tool kit for acceleration of any elliptic curve algorithm.

In general, the Pledge is expected to support only a single algorithm, while the Registrar, usually not constrained, is expected to support a wide variety of algorithms: both legacy ones and up-and-coming ones via regular software updates.

An example of the supported COSE\_Sign1 object structure is shown in Figure 3.

```
18( / COSE_Sign1 /
  [
    h'A101382E',          / protected header encoding: {1: -47}      /
    {                     /           which means { "alg": ES256K }    /
      4 : h'7890A03F1234' / 4 is the "kid" binary key identifier /
    },
    h'1234....5678', / voucher-request binary content (CBOR)      /
    h'4567....1234' / voucher-request binary public signature     /
  ]
)
```

Figure 3: COSE\_Sign1 example in CBOR diagnostic notation

The [COSE-registry] specifies the integers/encoding for the "alg" and "kid" fields in Figure 3. The "alg" field restricts the key usage for verification of this COSE object to a particular cryptographic algorithm.

The "kid" field is optionally present: it is an unprotected field that identifies the public key of the key pair that was used to sign this message. The value of the key identifier "kid" parameter is an example value. Usually a hash of the public key is used to identify the public key, but a device serial number may also be used. The choice of key identifier method is vendor-specific. If "kid" is not present, then a verifying party needs to use other context information to retrieve the right public key to verify the COSE\_Sign1 object against. For example, this context information may be a unique serial number encoded in the binary content (CBOR) field.

A Registrar MAY use a "kid" parameter in its RVR to identify its signing key as used to sign the RVR. The method of generating this "kid" is vendor-specific and SHOULD be configurable in the Registrar to support commonly used methods. In order to support future business cases and supply chain integrations, a Registrar MUST be configurable, on a per-manufacturer basis, to be able to configure the "kid" to a particular value. Both binary and string values are to be supported, each being inserted using a CBOR bstr or tstr. By default, a Registrar does not include a "kid" parameter in its RVR since the signing key is already identified by the included signing certificates in the COSE "x5bag" structure.

A Pledge normally SHOULD NOT use a "kid" parameter in its PVR, because its signing key is already identified by the Pledge's unique serial number that is included in the PVR. Still, where needed the Pledge MAY use a "kid" parameter in its PVR to help the MASA identify the right public key to verify against. This can occur for example if a Pledge has multiple IDevID identities. A Registrar normally SHOULD ignore a "kid" parameter used in a received PVR, as this information is intended for the MASA. In other words, there is no need for the Registrar to verify the contents of this field, but it may include it in an audit log.

In Appendix C a binary COSE\_Sign1 object is shown based on the voucher-request example of Section 9.1.4.

## 10. Deployment-specific Discovery Considerations

This section details how discovery is done in specific deployment scenarios.

### 10.1. 6TSCH Deployments

In 6TISCH networks, the Constrained Join Proxy (CoJP) mechanism is described in [RFC9031]. Such networks are expected to use a [I-D.ietf-lake-edhoc] to do key management. This is the subject of future work. The Enhanced Beacon has been extended in [RFC9032] to allow for discovery of the Join Proxy.

### 10.2. Generic networks using GRASP

[RFC8995] defines a mechanism for the Pledge to discover a Join Proxy by listening for [RFC8990] GRASP messages. This mechanism can be used on any network which does not have another more specific mechanism. This mechanism supports mesh networks, and can also be used over unencrypted WIFI.

### 10.3. Generic networks using mDNS

[RFC8995] also defines a non-normative mechanism for the Pledge to discover a Join Proxy by doing mDNS queries. This mechanism can be used on any network which does not have another recommended mechanism. This mechanism does not easily support mesh networks. It can be used over unencrypted WIFI.

### 10.4. Thread networks using Mesh Link Establishment (MLE)

Thread [Thread] is a wireless mesh network protocol based on 6LoWPAN [RFC6282] and other IETF protocols. In Thread, a new device discovers potential Thread networks and Thread routers to join by using the Mesh Link Establishment (MLE) [I-D.ietf-6lo-mesh-link-establishment] protocol. MLE uses the UDP port number 19788. The new device sends discovery requests on different IEEE 802.15.4 radio channels, to which routers (if any present) respond with a discovery response containing information about their respective network. Once a suitable router is selected the new device initiates a DTLS transport-layer secured connection to the network's commissioning application, over a link-local single radio hop to the selected Thread router. This link is not yet secured at the radio level: link-layer security will be set up once the new device is approved by the commissioning application to join the Thread network, and it gets provisioned with network access credentials.

The Thread router acts here as a Join Proxy. The MLE discovery response message contains UDP port information to signal the new device which port to use for its DTLS connection.

### 10.5. Non-mesh networks using CoAP Discovery

On unencrypted constrained networks such as 802.15.4, CoAP discover may be done using the mechanism detailed in [I-D.ietf-ace-coap-est] section 5.1.

## 11. Design Considerations

The design considerations for the CBOR encoding of vouchers are much the same as for JSON vouchers in [RFC8366]. One key difference is that the names of the leaves in the YANG definition do not affect the size of the resulting CBOR, as the SID translation process assigns integers to the names.

Any POST request to the Registrar with resource /vs or /es returns a 2.04 Changed response with empty payload. The client should be aware that the server may use a piggybacked CoAP response (ACK, 2.04) but may also respond with a separate CoAP response, i.e. first an (ACK, 0.0) that is an acknowledgement of the request reception followed by a (CON, 2.04) response in a separate CoAP message.

## 12. Raw Public Key Use Considerations

This section explains techniques to reduce the number of bytes that are sent over the wire during the BRSKI bootstrap. The use of a raw public key (RPK) in the pinning process can significantly reduce the number of bytes and round trips, but it comes with a few significant operational limitations.

### 12.1. The Registrar Trust Anchor

When the Pledge first connects to the Registrar, the connection to the Registrar is provisional, as explained in Section 5.6.2 of [RFC8995]. The Registrar provides its public key in a TLSServerCertificate, and the Pledge uses that to validate that integrity of the (D)TLS connection, but it does not validate the identity of the provided certificate.

As the TLSServerCertificate object is never verified directly by the pledge, sending it can be considered superfluous. Instead of using a (TLSServer)Certificate of type X509 (see section 4.4.2 of [RFC8446]), a RawPublicKey object is used.

A Registrar operating in a mixed environment can determine whether to send a Certificate or a Raw Public key: this is determined by the pledge including the server\_certificate\_type of RawPublicKey. This is shown in section 5 of [RFC7250].

The Pledge continues to send a `client_certificate_type` of X509, so that the Registrar can properly identify the pledge and distill the MASA URI information from its certificate.

#### 12.2. The Pledge Voucher Request

The Pledge puts the Registrar's public key into the `proximity-registrar-pubk` field of the `voucher-request`. (The `proximity-registrar-pubk-sha256` can also be used if the 32-bytes of a SHA256 hash turns out to be smaller than a typical ECDSA key.)

As the format of the `pubk` field is identical to the TLS Certificate `RawPublicKey`, no manipulation at all is needed to insert this into a `voucher-request`.

#### 12.3. The Voucher Response

A returned voucher will have a `pinned-domain-subk` field with the identical key as was found in the `proximity-registrar-pubk` field above, as well as in the TLS connection.

Validation of this key by the pledge is what takes the DTLS connection out of the provisional state see Section 5.6.2 of [RFC8995].

The voucher needs to be validated first. The Pledge needs to have a public key to validate the signature from the MASA on the voucher.

In certain cases, the MASA's public key counterpart of the (private) signing key is already installed in the Pledge at manufacturing time. In other cases, if the MASA signing key is based upon a PKI (see [I-D.richardson-anima-masa-considerations] Section 2.3), then a certificate chain may need to be included with the voucher in order for the pledge to validate the signature. In CMS signed artifacts, the CMS structure has a place for such certificates.

In the COSE-signed Constrained Vouchers described in this document, the `x5bag` attribute from [I-D.ietf-cose-x509] is to be used for this.

#### 13. Use of constrained vouchers with HTTPS

This specification contains two extensions to [RFC8995]: a constrained voucher format (COSE), and a constrained transfer protocol (CoAP).

On constrained networks with constrained devices, it make senses to use both together. However, this document does not mandate that this be the only way.

A given constrained device design and software may be re-used for multiple device models, such as a model having only an IEEE 802.15.4 radio, or a model having only an IEEE 802.11 (Wi-Fi) radio, or a model having both these radios. A manufacturer of such device models may wish to have code only for the use of the constrained voucher format (COSE), and use it on all supported radios including the IEEE 802.11 radio. For this radio, the software stack to support HTTP/TLS may be already integrated into the radio module hence it is attractive for the manufacturer to reuse this. This type of approach is supported by this document. In the case that HTTPS is used, the normal [RFC8995] resource names are used, together with the media types described in this document.

Other combinations are possible, but they are not enumerated here. New work such as [I-D.ietf-anima-jws-voucher] provides new formats that may be useable over a number of different transports. In general, sending larger payloads over constrained networks makes less sense, while sending smaller payloads over unconstrained networks is perfectly acceptable.

The Pledge will in most cases support a single voucher format, which it uses without negotiation i.e. without discovery of formats supported. The Registrar, being unconstrained, is expected to support all voucher formats. There will be cases where a Registrar does not support a new format that a new Pledge uses, and this is an unfortunate situation that will result in lack of interoperation.

The responsibility for supporting new formats is on the Registrar.

## 14. Security Considerations

### 14.1. Duplicate serial-numbers

In the absense of correct use of idevid-issuer by the Registrar as detailed in Section 8.4, it would be possible for a malicious Registrar to use an unauthorized voucher for a device. This would apply only to the case where a Manufacturer Authorized Signing Authority (MASA) is trusted by different products from the same manufacturer, and the manufacturer has duplicated serial numbers as a result of a merge, acquisition or mis-management.

For example, imagine the same manufacturer makes light bulbs as well as gas centrifuges, and said manufacturer does not uniquely allocate product serial numbers. This attack only works for nonceless vouchers. The attacker has obtained a light bulb which happens to have the same serial-number as a gas centrifuge which it wishes to obtain access. The attacker performs a normal BRSKI onboarding for the light bulb, but then uses the resulting voucher to onboard the

gas centrifuge. The attack requires that the gas centrifuge be returned to a state where it is willing to perform a new onboarding operation.

This attack is prevented by the mechanism of having the Registrar include the idevid-issuer in the RVR, and the MASA including it in the resulting voucher. The idevid-issuer is not included by default: a MASA needs to be aware if there are parts of the organization which duplicates serial numbers, and if so, include it.

#### 14.2. IDevID security in Pledge

The security of this protocol depends upon the Pledge identifying itself to the Registrar using its manufacturer installed certificate: the IDevID certificate. Associated with this certificate is the IDevID private key, known only to the Pledge. Disclosure of this private key to an attacker would permit the attacker to impersonate the Pledge towards the Registrar, probably gaining access credentials to that Registrar's network.

If the IDevID private key disclosure is known to the manufacturer, there is little recourse other than recall of the relevant part numbers. The process for communicating this recall would be within the BRSKI-MASA protocol. Neither this specification nor [RFC8995] provides for consultation of a Certification Revocation List (CRL) or Open Certificate Status Protocol (OCSP) by a Registrar when evaluating an IDevID certificate. However, the BRSKI-MASA protocol submits the IDevID from the Registrar to the manufacturer's MASA and a manufacturer would have an opportunity to decline to issue a voucher for a device which they believe has become compromised.

It may be difficult for a manufacturer to determine when an IDevID private key has been disclosed. Two situations present themselves: in the first situation a compromised private key might be reused in a counterfeit device, which is sold to another customer. This would present itself as an onboarding of the same device in two different networks. The manufacturer may become suspicious seeing two voucher requests for the same device from different Registrars. Such activity could be indistinguishable from a device which has been resold from one operator to another, or re-deployed by an operator from one location to another.

In the second situation, an attacker having compromised the IDevID private key of a device might then install malware into the same device and attempt to return it to service. The device, now blank, would go through a second onboarding process with the original Registrar. Such a Registrar could notice that the device has been "factory reset" and alert the operator to this situation. One remedy

against the presence of malware is through the use of Remote Attestation such as described in [I-D.ietf-rats-architecture]. Future work will need to specify a background-check Attestation flow as part of the voucher-request/voucher-response process. Attestation may still require access to a private key (e.g. IDevID private key) in order to sign Evidence, so a primary goal should be to keep any private key safe within the Pledge.

In larger, more expensive, systems there is budget (power, space, and bill of materials) to include more specific defenses for a private key. For instance, this includes putting the IDevID private key in a Trusted Platform Module (TPM), or use of Trusted Execution Environments (TEE) for access to the key. On smaller IoT devices, the cost and power budget for an extra part is often prohibitive.

It is becoming more and more common for CPUs to have an internal set of one-time fuses that can be programmed (often they are "burnt" by a laser) at the factory. This section of memory is only accessible in some privileged CPU state. The use of this kind of CPU is appropriate as it provides significant resistance against key disclosure even when the device can be disassembled by an attacker.

In a number of industry verticals, there is increasing concern about counterfeit parts. These may be look-alike parts created in a different factory, or parts which are created in the same factory during an illegal night-shift, but which are not subject to the appropriate level of quality control. The use of a manufacturer-signed IDevID certificate provides for discovery of the pedigree of each part, and this often justifies the cost of the security measures associated with storing the private key.

#### 14.3. Security of CoAP and UDP protocols

Section 7.1 explains that no CoAPS version of the BRSKI-MASA protocol is proposed. The connection from the Registrar to the MASA continues to be HTTPS as in [RFC8995]. This has been done to simplify the MASA deployment for the manufacturer, because no new protocol needs to be enabled on the server.

The use of UDP protocols across the open Internet is sometimes fraught with security challenges. Denial-of-service attacks against UDP based protocols are trivial as there is no three-way handshake as done for TCP. The three-way handshake of TCP guarantees that the node sending the connection request is reachable using the origin IP address. While DTLS contains an option to do a stateless challenge -- a process actually stronger than that done by TCP -- it is not yet common for this mechanism to be available in hardware at multigigabit speeds. It is for this reason that this document defines using HTTPS for the Registrar to MASA connection.

#### 14.4. Registrar Certificate may be self-signed

The provisional (D)TLS connection formed by the Pledge with the Registrar does not authenticate the Registrar's identity. This Registrar's identity is validated by the [RFC8366] voucher that is issued by the MASA, signed with an anchor that was built-in to the Pledge.

The Registrar may therefore use any certificate, including a self-signed one. The only restrictions on the certificate is that it **MUST** have EKU bits set as detailed in Section 7.3.1.

#### 14.5. Use of RPK alternatives to proximity-registrar-cert

In Section 9.1 two compact alternative fields for proximity-registrar-cert are defined that include an RPK: proximity-registrar-pubk and proximity-registrar-pubk-sha256. The Pledge can use these fields in its PVR to identify the Registrar based on its public key only. Since the full certificate of the proximate Registrar is not included, use of these fields by a Pledge implies that a Registrar could insert another certificate with the same public key identity into the RVR. For example, an older or a newer version of its certificate. The MASA will not be able to detect such act by the Registrar. But since any 'other' certificate the Registrar could insert in this way still encodes its identity the additional risk of using the RPK alternatives is negligible.

When a Registrar sees a PVR that uses one of proximity-registrar-pubk or proximity-registrar-pubk-sha256 fields, this implies the Registrar must include the certificate identified by these fields into its RVR. Otherwise, the MASA is unable to verify proximity. This requirement is already implied by the "MUST" requirement in Section 8.1.

#### 14.6. MASA support of CoAPS

The use of CoAP for the BRSKI-MASA connection is not in scope of the current document. The following security considerations have led to this choice of scope:

- \* the technology and experience to build secure Internet-scale HTTPS responders (which the MASA is) is common, while the experience in doing the same for CoAP is much less common.
- \* in many enterprise networks, outgoing UDP connections are often treated as suspicious, which could effectively block CoAP connections for some firewall configurations.
- \* reducing the complexity of MASA (i.e. less protocols supported) would also reduce its potential attack surface, which is relevant since the MASA is 24/7 exposed on the Internet and accepting (untrusted) incoming connections.

#### 15. IANA Considerations

##### 15.1. Resource Type Registry

Additions to the sub-registry "Resource Type Link Target Attribute Values", within the "CoRE Parameters" IANA registry are specified below.

Reference: [This RFC]

Attribute	Description
brski	Root path of Bootstrapping Remote Secure Key Infrastructure (BRSKI) resources
brski.rv	BRSKI request voucher resource
brski.vs	BRSKI voucher status telemetry resource
brski.es	BRSKI enrollment status telemetry resource

Table 4: Resource Type (rt) link target attribute values for IANA registration

## 15.2. The IETF XML Registry

This document registers two URIs in the IETF XML registry [RFC3688]. Following the format in [RFC3688], the following registration is requested:

URI: urn:ietf:params:xml:ns:yang:ietf-voucher-constrained  
 Registrant Contact: The ANIMA WG of the IETF.  
 XML: N/A, the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-voucher-request-constrained  
 Registrant Contact: The ANIMA WG of the IETF.  
 XML: N/A, the requested URI is an XML namespace.

## 15.3. The YANG Module Names Registry

This document registers two YANG modules in the YANG Module Names registry [RFC6020]. Following the format defined in [RFC6020], the the following registration is requested:

name: ietf-voucher-constrained  
 namespace: urn:ietf:params:xml:ns:yang:ietf-voucher-constrained  
 prefix: vch  
 reference: RFC XXXX

name: ietf-voucher-request-constrained  
 namespace: urn:ietf:params:xml:ns:yang:ietf-voucher-request-constrained  
 prefix: vch  
 reference: RFC XXXX

## 15.4. The RFC SID range assignment sub-registry

Entry-point	Size	Module name	RFC Number
2450	50	ietf-voucher-constrained	[This RFC]
2500	50	ietf-voucher-request -constrained	[This RFC]

Warning: These SID values are defined in [I-D.ietf-core-sid], not as an Early Allocation.

IANA: please update the names in the Registry to match these revised names, if they have not already been revised.

### 15.5. Media Types Registry

This section registers the 'application/voucher-cose+cbor' in the IANA "Media Types" registry. This media type is used to indicate that the content is a CBOR voucher or voucher request signed with a COSE\_Sign1 structure [I-D.ietf-cose-rfc8152bis-struct].

#### 15.5.1. application/voucher-cose+cbor

Type name: application  
Subtype name: voucher-cose+cbor  
Required parameters: N/A  
Optional parameters: N/A  
Encoding considerations: binary (CBOR)  
Security considerations: Security Considerations of [This RFC].  
Interoperability considerations: The format is designed to be broadly interoperable.  
Published specification: [This RFC]  
Applications that use this media type: ANIMA, 6tisch, and other zero-touch onboarding systems  
Fragment identifier considerations: The syntax and semantics of fragment identifiers specified for application/voucher-cose+cbor are as specified for application/cbor. (At publication of this document, there is no fragment identification syntax defined for application/cbor.)  
Additional information:  
 Deprecated alias names for this type: N/A  
 Magic number(s): N/A  
 File extension(s): .vch  
 Macintosh file type code(s): N/A  
Person & email address to contact for further information: IETF ANIMA Working Group (anima@ietf.org) or IETF Operations and Management Area Working Group (opsawg@ietf.org)  
Intended usage: COMMON  
Restrictions on usage: N/A  
Author: ANIMA WG  
Change controller: IETF  
Provisional registration? (standards tree only): NO

### 15.6. CoAP Content-Format Registry

One addition to the sub-registry "CoAP Content-Formats", within the "CoRE Parameters" registry is needed for a new content-format. It can be registered in the Expert Review range (0-255) or the IETF Review range (256-9999).

Media type	Encoding	ID	Reference
application/voucher-cose+cbor	-	TBD3	[This RFC]

### 15.7. Update to BRSKI Parameters Registry

This section updates the BRSKI Well-Known URIs sub-registry of the IANA Bootstrapping Remote Secure Key Infrastructures (BRSKI) Parameters Registry by adding a new column "Short URI". The contents of this field MUST be specified for any newly registered URI as follows:

Short URI: A short name for the "URI" resource that can be used by a Constrained BRSKI Pledge in a CoAP request to the Registrar. In case the "URI" resource is only used between Registrar and MASA, the value "--" is registered denoting that a short name is not applicable.

The initial contents of the sub-registry including the new column are as follows:

URI	Short URI	Description	Reference
requestvoucher	rv	Request voucher: Pledge to Registrar, and Registrar to MASA	[RFC8995], [This RFC]
voucher_status	vs	Voucher status telemetry: Pledge to Registrar	[RFC8995], [This RFC]
requestauditlog	--	Request audit log: Registrar to MASA	[RFC8995]
enrollstatus	es	Enrollment status telemetry: Pledge to Registrar	[RFC8995], [This RFC]

Table 5: Update of the BRSKI Well-Known URI Sub-Registry

## 16. Acknowledgements

We are very grateful to Jim Schaad for explaining COSE and CMS choices. Also thanks to Jim Schaad for correcting earlier versions of the COSE\_Sign1 objects.

Michel Veillette did extensive work on `_pyang_` to extend it to support the SID allocation process, and this document was among its first users.

Daniel Franke and Henk Birkholtz provided review feedback.

The BRSKI design team has met on many Thursdays for document review. It includes: Aurelio Schellenbaum, David von Oheimb Steffen Fries, Thomas Werner, Toerless Eckert,

## 17. Changelog

-11 to -16 (For change details see GitHub issues <https://github.com/anima-wg/constrained-voucher/issues>)

-10 Design considerations extended Examples made consistent

-08 Examples for cose\_sign1 are completed and improved.

-06 New SID values assigned; regenerated examples

-04 voucher and request-voucher MUST be signed examples for signed request are added in appendix IANA SID registration is updated SID values in examples are aligned signed cms examples aligned with new SIDs

-03

Examples are inverted.

-02

Example of requestvoucher with unsigned application/cbor is added attributes of voucher "refined" to optional CBOR serialization of vouchers improved Discovery port numbers are specified

-01

application/json is optional, application/cbor is compulsory Cms and cose mediatypes are introduced

## 18. References

### 18.1. Normative References

[I-D.ietf-ace-coap-est]

Stok, P. V. D., Kampanakis, P., Richardson, M. C., and S. Raza, "EST over secure CoAP (EST-coaps)", Work in Progress, Internet-Draft, draft-ietf-ace-coap-est-18, 6 January 2020, <<https://www.ietf.org/archive/id/draft-ietf-ace-coap-est-18.txt>>.

[I-D.ietf-core-sid]

Veillette, M., Pelov, A., Petrov, I., Bormann, C., and M. Richardson, "YANG Schema Item iDentifier (YANG SID)", Work in Progress, Internet-Draft, draft-ietf-core-sid-18, 18 November 2021, <<https://www.ietf.org/archive/id/draft-ietf-core-sid-18.txt>>.

[I-D.ietf-core-yang-cbor]

Veillette, M., Petrov, I., Pelov, A., Bormann, C., and M. Richardson, "CBOR Encoding of Data Modeled with YANG", Work in Progress, Internet-Draft, draft-ietf-core-yang-cbor-19, 20 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-core-yang-cbor-19.txt>>.

[I-D.ietf-cose-rfc8152bis-algs]

Schaad, J., "CBOR Object Signing and Encryption (COSE): Initial Algorithms", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-algs-12, 24 September 2020, <<https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-algs-12.txt>>.

[I-D.ietf-cose-rfc8152bis-struct]

Schaad, J., "CBOR Object Signing and Encryption (COSE): Structures and Process", Work in Progress, Internet-Draft, draft-ietf-cose-rfc8152bis-struct-15, 1 February 2021, <<https://www.ietf.org/archive/id/draft-ietf-cose-rfc8152bis-struct-15.txt>>.

[I-D.ietf-cose-x509]

Schaad, J., "CBOR Object Signing and Encryption (COSE): Header parameters for carrying and referencing X.509 certificates", Work in Progress, Internet-Draft, draft-ietf-cose-x509-08, 14 December 2020, <<https://www.ietf.org/archive/id/draft-ietf-cose-x509-08.txt>>.

- [I-D.ietf-tls-dtls13]  
Rescorla, E., Tschofenig, H., and N. Modadugu, "The Datagram Transport Layer Security (DTLS) Protocol Version 1.3", Work in Progress, Internet-Draft, draft-ietf-tls-dtls13-43, 30 April 2021, <<https://www.ietf.org/archive/id/draft-ietf-tls-dtls13-43.txt>>.
- [ieee802-1AR]  
IEEE Standard, "IEEE 802.1AR Secure Device Identifier", 2009, <<http://standards.ieee.org/findstds/standard/802.1AR-2009.html>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <<https://www.rfc-editor.org/info/rfc3688>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC4210] Adams, C., Farrell, S., Kause, T., and T. Mononen, "Internet X.509 Public Key Infrastructure Certificate Management Protocol (CMP)", RFC 4210, DOI 10.17487/RFC4210, September 2005, <<https://www.rfc-editor.org/info/rfc4210>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.

- [RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <<https://www.rfc-editor.org/info/rfc6066>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J., Weiler, S., and T. Kivinen, "Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250, June 2014, <<https://www.rfc-editor.org/info/rfc7250>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <<https://www.rfc-editor.org/info/rfc8366>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.
- [RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/info/rfc8949>>.

- [RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <<https://www.rfc-editor.org/info/rfc8995>>.
- [RFC9031] Vuini, M., Ed., Simon, J., Pister, K., and M. Richardson, "Constrained Join Protocol (CoJP) for 6TiSCH", RFC 9031, DOI 10.17487/RFC9031, May 2021, <<https://www.rfc-editor.org/info/rfc9031>>.
- [RFC9032] Dujovne, D., Ed. and M. Richardson, "Encapsulation of 6TiSCH Join and Enrollment Information Elements", RFC 9032, DOI 10.17487/RFC9032, May 2021, <<https://www.rfc-editor.org/info/rfc9032>>.

## 18.2. Informative References

- [COSE-registry]  
IANA, "CBOR Object Signing and Encryption (COSE) registry", 2017, <<https://www.iana.org/assignments/cose/cose.xhtml>>.
- [I-D.ietf-6lo-mesh-link-establishment]  
Kelsey, R., "Mesh Link Establishment", Work in Progress, Internet-Draft, draft-ietf-6lo-mesh-link-establishment-00, 1 December 2015, <<https://www.ietf.org/archive/id/draft-ietf-6lo-mesh-link-establishment-00.txt>>.
- [I-D.ietf-anima-constrained-join-proxy]  
Richardson, M., Stok, P. V. D., and P. Kampanakis, "Constrained Join Proxy for Bootstrapping Protocols", Work in Progress, Internet-Draft, draft-ietf-anima-constrained-join-proxy-09, 25 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-anima-constrained-join-proxy-09.txt>>.
- [I-D.ietf-anima-jws-voucher]  
Richardson, M. and T. Werner, "JWS signed Voucher Artifacts for Bootstrapping Protocols", Work in Progress, Internet-Draft, draft-ietf-anima-jws-voucher-03, 7 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-anima-jws-voucher-03.txt>>.

- [I-D.ietf-lake-edhoc]  
Selander, G., Mattsson, J. P., and F. Palombini,  
"Ephemeral Diffie-Hellman Over COSE (EDHOC)", Work in  
Progress, Internet-Draft, draft-ietf-lake-edhoc-12, 20  
October 2021, <<https://www.ietf.org/archive/id/draft-ietf-lake-edhoc-12.txt>>.
- [I-D.ietf-rats-architecture]  
Birkholz, H., Thaler, D., Richardson, M., Smith, N., and  
W. Pan, "Remote Attestation Procedures Architecture", Work  
in Progress, Internet-Draft, draft-ietf-rats-architecture-  
15, 8 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-rats-architecture-15.txt>>.
- [I-D.kuehlewind-update-tag]  
Kuehlewind, M. and S. Krishnan, "Definition of new tags  
for relations between RFCs", Work in Progress, Internet-  
Draft, draft-kuehlewind-update-tag-04, 12 July 2021,  
<<https://www.ietf.org/archive/id/draft-kuehlewind-update-tag-04.txt>>.
- [I-D.richardson-anima-masa-considerations]  
Richardson, M. and W. Pan, "Operational Considerations for  
Voucher infrastructure for BRSKI MASA", Work in Progress,  
Internet-Draft, draft-richardson-anima-masa-  
considerations-06, 13 November 2021,  
<<https://www.ietf.org/archive/id/draft-richardson-anima-masa-considerations-06.txt>>.
- [RFC4443] Conta, A., Deering, S., and M. Gupta, Ed., "Internet  
Control Message Protocol (ICMPv6) for the Internet  
Protocol Version 6 (IPv6) Specification", STD 89,  
RFC 4443, DOI 10.17487/RFC4443, March 2006,  
<<https://www.rfc-editor.org/info/rfc4443>>.
- [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6  
Datagrams over IEEE 802.15.4-Based Networks", RFC 6282,  
DOI 10.17487/RFC6282, September 2011,  
<<https://www.rfc-editor.org/info/rfc6282>>.
- [RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link  
Format", RFC 6690, DOI 10.17487/RFC6690, August 2012,  
<<https://www.rfc-editor.org/info/rfc6690>>.
- [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed.,  
"Enrollment over Secure Transport", RFC 7030,  
DOI 10.17487/RFC7030, October 2013,  
<<https://www.rfc-editor.org/info/rfc7030>>.

- [RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <<https://www.rfc-editor.org/info/rfc7228>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.
- [RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <<https://www.rfc-editor.org/info/rfc8990>>.
- [Thread] Thread Group, Inc, "Thread support page, White Papers", November 2021, <<https://www.threadgroup.org/support#Whitepapers>>.

#### Appendix A. Library Support for BRSKI

For the implementation of BRSKI, the use of a software library to manipulate certificates and use crypto algorithms is often beneficial. Two C-based examples are OpenSSL and mbedtls. Others more targeted to specific platforms or languages exist. It is important to realize that the library interfaces differ significantly between libraries.

Libraries do not support all known crypto algorithms. Before deciding on a library, it is important to look at their supported crypto algorithms and the roadmap for future support. Apart from availability, the library footprint, and the required execution cycles should be investigated beforehand.

The handling of certificates usually includes the checking of a certificate chain. In some libraries, chains are constructed and verified on the basis of a set of certificates, the trust anchor (usually self signed root CA), and the target certificate. In other libraries, the chain must be constructed beforehand and obey order criteria. Verification always includes the checking of the signatures. Less frequent is the checking the validity of the dates or checking the existence of a revoked certificate in the chain against a set of revoked certificates. Checking the chain on the consistency of the certificate extensions which specify the use of the certificate usually needs to be programmed explicitly.

A library can be used to construct a (D)TLS connection. It is useful to realize that differences between (D)TLS implementations will occur due to the differences in the certificate checks supported by the

library. On top of that, checks between client and server certificates enforced by (D)TLS are not always helpful for a BRSKI implementation. For example, the certificates of Pledge and Registrar are usually not related when the BRSKI protocol is started. It must be verified that checks on the relation between client and server certificates do not hamper a successful DTLS connection establishment.

#### A.1. OpensSSL

From openssl's apps/verify.c :

```
<CODE BEGINS>
X509 *x = NULL;
int i = 0, ret = 0;
X509_STORE_CTX *csc;
STACK_OF(X509) *chain = NULL;
int num_untrusted;

x = load_cert(file, "certificate file");
if (x == NULL)
    goto end;

csc = X509_STORE_CTX_new();
if (csc == NULL) {
    BIO_printf(bio_err, "error %s: X.509 store context"
               "allocation failed\n",
               (file == NULL) ? "stdin" : file);
    goto end;
}

X509_STORE_set_flags(ctx, vflags);
if (!X509_STORE_CTX_init(csc, ctx, x, uchain)) {
    X509_STORE_CTX_free(csc);
    BIO_printf(bio_err,
               "error %s: X.509 store context"
               "initialization failed\n",
               (file == NULL) ? "stdin" : file);
    goto end;
}
if (tchain != NULL)
    X509_STORE_CTX_set0_trusted_stack(csc, tchain);
if (crls != NULL)
    X509_STORE_CTX_set0_crls(csc, crls);

i = X509_verify_cert(csc);
X509_STORE_CTX_free(csc);

<CODE ENDS>
```

#### A.2. mbedTLS

```
<CODE BEGINS>
mbedtls_x509_crt cert;
mbedtls_x509_crt caCert;
uint32_t certVerifyResultFlags;
...
int result = mbedtls_x509_crt_verify(&cert, &caCert, NULL, NULL,
                                     &certVerifyResultFlags, NULL, NULL);
<CODE ENDS>
```

## Appendix B. Constrained BRSKI-EST Message Examples

This appendix extends the message examples from Appendix A of [I-D.ietf-ace-coap-est] with constrained BRSKI messages. The CoAP headers are only fully worked out for the first example, enrollstatus.

## B.1. enrollstatus

A coaps enrollstatus message from Pledge to Registrar can be as follows:

```
POST coaps://192.0.2.1:8085/b/es
Content-Format: 60
Payload: <binary CBOR enrollstatus document>
```

The corresponding CoAP header fields are shown below.

```
Ver = 1
T = 0 (CON)
TKL = 1
Code = 0x02 (0.02 is POST method)
Message ID = 0xab0f
Token = 0x4d
Options
  Option (Uri-Path)
    Option Delta = 0xb (option nr = 11)
    Option Length = 0x1
    Option Value = "b"
  Option (Uri-Path)
    Option Delta = 0x0 (option nr = 11)
    Option Length = 0x2
    Option Value = "es"
  Option (Content-Format)
    Option Delta = 0x1 (option nr = 12)
    Option Length = 0x1
    Option Value = 60 (application/cbor)
Payload Marker = 0xFF
Payload = A26776657273696F6E0166737461747573F5 (18 bytes binary)
```

The Uri-Host and Uri-Port Options are omitted because they coincide with the transport protocol (UDP) destination address and port respectively.

The above binary CBOR enrollstatus payload looks as follows in CBOR diagnostic notation, for the case of enrollment success:

```
{
  "version": 1,
  "status": true
}
```

Alternatively the payload could look as follows in case of enrollment failure, using the reason field to describe the failure:

```
Payload = A36776657273696F6E0166737461747573F466726561736F6E782A3C
          496E666F726D61746976652068756D616E207265616461626C652065
          72726F72206D6573736167653E
```

```
{
  "version": 1,
  "status": false,
  "reason": "<Informative human readable error message>"
}
```

To indicate successful reception of the enrollmentstatus telemetry report, a response from the Registrar may then be:

2.04 Changed

With CoAP fields:

```
Ver=1
T=2 (ACK)
TKL=1
Code = 0x44 (2.04 Changed)
Message ID = 0xab0f
Token = 0x4d
```

## B.2. voucher\_status

A coaps voucher\_status message from Pledge to Registrar can be as follows:

```
POST coaps://[2001:db8::2:1]/.well-known/brski/vs
Content-Format: 60 (application/cbor)
Payload:
a46776657273696f6e0166737461747573f466726561736f6e7828496e66
6f726d61746976652068756d616e2d7265616461626c65206572726f7220
6d6573736167656e726561736f6e2d636f6e74657874a100764164646974
696f6e616c20696e666f726d6174696f6e
```

The request payload above is binary CBOR but represented here in hexadecimal for readability. Below is the equivalent CBOR diagnostic format.

```
{
  "version": 1,
  "status": false,
  "reason": "Informative human-readable error message",
  "reason-context": { 0: "Additional information" }
}
```

A success response without payload will then be sent by the Registrar back to the Pledge to indicate reception of the telemetry report:

#### 2.04 Changed

### Appendix C. COSE-signed Voucher (Request) Examples

This appendix provides examples of COSE-signed voucher requests and vouchers. First, the used test keys and certificates are described, following by examples of a constrained PVR, RVR and voucher.

#### C.1. Pledge, Registrar and MASA Keys

This section documents the public and private keys used for all examples in this appendix. These keys are not used in any production system, and must only be used for testing purposes.

##### C.1.1. Pledge IDevID private key

```
<CODE BEGINS>
Private-Key: (256 bit)
priv:
  9b:4d:43:b6:a9:e1:7c:04:93:45:c3:13:d9:b5:f0:
  41:a9:6a:9c:45:79:73:b8:62:f1:77:03:3a:fc:c2:
  9c:9a
pub:
  04:d6:b7:6f:74:88:bd:80:ae:5f:28:41:2c:72:02:
  ef:5f:98:b4:81:e1:d9:10:4c:f8:1b:66:d4:3e:5c:
  ea:da:73:e6:a8:38:a9:f1:35:11:85:b6:cd:e2:04:
  10:be:fe:d5:0b:3b:14:69:2e:e1:b0:6a:bc:55:40:
  60:eb:95:5c:54
ASN1 OID: prime256v1
NIST CURVE: P-256
<CODE ENDS>
```

##### C.1.2. Registrar private key

```

<CODE BEGINS>
Private-Key: (256 bit)
priv:
  81:df:bb:50:a3:45:58:06:b5:56:3b:46:de:f3:e9:
  e9:00:ae:98:13:9e:2f:36:68:81:fc:d9:65:24:fb:
  21:7e
pub:
  04:50:7a:c8:49:1a:8c:69:c7:b5:c3:1d:03:09:ed:
  35:ba:13:f5:88:4c:e6:2b:88:cf:30:18:15:4f:a0:
  59:b0:20:ec:6b:eb:b9:4e:02:b8:93:40:21:89:8d:
  a7:89:c7:11:ce:a7:13:39:f5:0e:34:8e:df:0d:92:
  3e:d0:2d:c7:b7
ASN1 OID: prime256v1
NIST CURVE: P-256
<CODE ENDS>

```

### C.1.3. MASA private key

```

<CODE BEGINS>
Private-Key: (256 bit)
priv:
  c6:bb:a5:8f:b6:d3:c4:75:28:d8:d3:d9:46:c3:31:
  83:6d:00:0a:9a:38:ce:22:5c:e9:d9:ea:3b:98:32:
  ec:31
pub:
  04:59:80:94:66:14:94:20:30:3c:66:08:85:55:86:
  db:e7:d4:d1:d7:7a:d2:a3:1a:0c:73:6b:01:0d:02:
  12:15:d6:1f:f3:6e:c8:d4:84:60:43:3b:21:c5:83:
  80:1e:fc:e2:37:85:77:97:94:d4:aa:34:b5:b6:c6:
  ed:f3:17:5c:f1
ASN1 OID: prime256v1
NIST CURVE: P-256
<CODE ENDS>

```

## C.2. Pledge, Registrar and MASA Certificates

All keys and certificates used for the examples have been generated with OpenSSL – see Appendix D for more details on certificate generation. Below the certificates are listed that accompany the keys shown above. Each certificate description is followed by the hexadecimal representation of the X.509 ASN.1 DER encoded certificate. This representation can be for example decoded using an online ASN.1 decoder.

### C.2.1. Pledge IDevID Certificate

<CODE BEGINS>

Certificate:

Data:

Version: 3 (0x2)  
Serial Number: 4822678189204992 (0x11223344556600)  
Signature Algorithm: ecdsa-with-SHA256  
Issuer: C=NL, ST=NB, L=Helmond, O=vanderstok, OU=manufacturer,  
CN=masa.stok.nl  
Validity  
Not Before: Dec 9 10:02:36 2020 GMT  
Not After : Dec 31 23:59:59 9999 GMT  
Subject: C=NL, ST=NB, L=Helmond, O=vanderstok, OU=manufacturing,  
CN=uuid:pledge.1.2.3.4/serialNumber=pledge.1.2.3.4  
Subject Public Key Info:  
Public Key Algorithm: id-ecPublicKey  
Public-Key: (256 bit)  
pub:  
04:d6:b7:6f:74:88:bd:80:ae:5f:28:41:2c:72:02:  
ef:5f:98:b4:81:e1:d9:10:4c:f8:1b:66:d4:3e:5c:  
ea:da:73:e6:a8:38:a9:f1:35:11:85:b6:cd:e2:04:  
10:be:fe:d5:0b:3b:14:69:2e:e1:b0:6a:bc:55:40:  
60:eb:95:5c:54  
ASN1 OID: prime256v1  
NIST CURVE: P-256  
X509v3 extensions:  
X509v3 Basic Constraints:  
CA:FALSE  
X509v3 Authority Key Identifier:  
keyid:  
E4:03:93:B4:C3:D3:F4:2A:80:A4:77:18:F6:96:49:03:01:17:68:A3

Signature Algorithm: ecdsa-with-SHA256  
30:46:02:21:00:d2:e6:45:3b:b0:c3:00:b3:25:8d:f1:83:fe:  
d9:37:c1:a2:49:65:69:7f:6b:b9:ef:2c:05:07:06:31:ac:17:  
bd:02:21:00:e2:ce:9e:7b:7f:74:50:33:ad:9e:ff:12:4e:e9:  
a6:f3:b8:36:65:ab:7d:80:bb:56:88:bc:03:1d:e5:1e:31:6f

<CODE ENDS>

Below is the hexadecimal representation:

<CODE BEGINS>

```
30820226308201cba003020102020711223344556600300a06082a8648ce3d04
0302306f310b3009060355040613024e4c310b300906035504080c024e423110
300e06035504070c0748656c6d6f6e6431133011060355040a0c0a76616e6465
7273746f6b31153013060355040b0c0c6d616e756666163747572657231153013
06035504030c0c6d6173612e73746f6b2e6e6c3020170d323031323039313030
3233365a180f39393939313233313233353935395a308190310b300906035504
0613024e4c310b300906035504080c024e423110300e06035504070c0748656c
6d6f6e6431133011060355040a0c0a76616e64657273746f6b31163014060355
040b0c0d6d616e756666163747572696e67311c301a06035504030c1375756964
3a706c656467652e312e322e332e34311730150603550405130e706c65646765
2e312e322e332e343059301306072a8648ce3d020106082a8648ce3d03010703
420004d6b76f7488bd80ae5f28412c7202ef5f98b481e1d9104cf81b66d43e5c
eada73e6a838a9f1351185b6cde20410befed50b3b14692ee1b06abc554060eb
955c54a32e302c30090603551d1304023000301f0603551d23041830168014e4
0393b4c3d3f42a80a47718f6964903011768a3300a06082a8648ce3d04030203
49003046022100d2e6453bb0c300b3258df183fed937c1a24965697f6bb9ef2c
05070631ac17bd022100e2ce9e7b7f745033ad9eff124ee9a6f3b83665ab7d80
bb5688bc031de51e316f
```

<CODE ENDS>

#### C.2.2. Registrar Certificate

<CODE BEGINS>

Certificate:

Data:

Version: 3 (0x2)

Serial Number:

70:56:ea:aa:30:66:d8:82:6a:55:5b:90:88:d4:62:bf:9c:f2:8c:fd

Signature Algorithm: ecdsa-with-SHA256

Issuer: C=NL, ST=NB, L=Helmond, O=vanderstok, OU=consultancy,  
CN=registrar.stok.nl

Validity

Not Before: Dec 9 10:02:36 2020 GMT

Not After : Dec 9 10:02:36 2021 GMT

Subject: C=NL, ST=NB, L=Helmond, O=vanderstok, OU=consultancy,  
CN=registrar.stok.nl

Subject Public Key Info:

Public Key Algorithm: id-ecPublicKey

Public-Key: (256 bit)

pub:

04:50:7a:c8:49:1a:8c:69:c7:b5:c3:1d:03:09:ed:  
35:ba:13:f5:88:4c:e6:2b:88:cf:30:18:15:4f:a0:  
59:b0:20:ec:6b:eb:b9:4e:02:b8:93:40:21:89:8d:  
a7:89:c7:11:ce:a7:13:39:f5:0e:34:8e:df:0d:92:  
3e:d0:2d:c7:b7

ASN1 OID: prime256v1

NIST CURVE: P-256

X509v3 extensions:

X509v3 Subject Key Identifier:

08:C2:BF:36:88:7F:79:41:21:85:87:2F:16:A7:AC:A6:EF:B3:D2:B3

X509v3 Authority Key Identifier:

keyid:

08:C2:BF:36:88:7F:79:41:21:85:87:2F:16:A7:AC:A6:EF:B3:D2:B3

X509v3 Basic Constraints: critical

CA:TRUE

X509v3 Extended Key Usage:

CMC Registration Authority, TLS Web Server

Authentication, TLS Web Client Authentication

X509v3 Key Usage: critical

Digital Signature, Non Repudiation, Key Encipherment,

Data Encipherment, Certificate Sign, CRL Sign

Signature Algorithm: ecdsa-with-SHA256

30:44:02:20:74:4c:99:00:85:13:b2:f1:bc:fd:f9:02:1a:46:

fb:17:4c:f8:83:a2:7c:a1:d9:3f:ae:ac:f3:1e:4e:dd:12:c6:

02:20:11:47:14:db:f5:1a:5e:78:f5:81:b9:42:1c:6e:47:02:

ab:53:72:70:c5:ba:fb:2d:16:c3:de:9a:a1:82:c3:5f

<CODE ENDS>

Below is the hexadecimal representation which is in (request-)voucher examples referred to as regis-cert-hex:

```
<CODE BEGINS>
308202753082021ca00302010202147056eaaa3066d8826a555b9088d462bf9c
f28cfd300a06082a8648ce3d0403023073310b3009060355040613024e4c310b
300906035504080c024e423110300e06035504070c0748656c6d6f6e64311330
11060355040a0c0a76616e64657273746f6b31143012060355040b0c0b636f6e
73756c74616e6379311a301806035504030c117265676973747261722e73746f
6b2e6e6c301e170d3230313230393130303233365a170d323131323039313030
3233365a3073310b3009060355040613024e4c310b300906035504080c024e42
3110300e06035504070c0748656c6d6f6e6431133011060355040a0c0a76616e
64657273746f6b31143012060355040b0c0b636f6e73756c74616e6379311a30
1806035504030c117265676973747261722e73746f6b2e6e6c3059301306072a
8648ce3d020106082a8648ce3d03010703420004507ac8491a8c69c7b5c31d03
09ed35ba13f5884ce62b88cf3018154fa059b020ec6bebb94e02b8934021898d
a789c711cea71339f50e348edf0d923ed02dc7b7a3818d30818a301d0603551d
0e0416041408c2bf36887f79412185872f16a7aca6efb3d2b3301f0603551d23
04183016801408c2bf36887f79412185872f16a7aca6efb3d2b3300f0603551d
130101fff040530030101fff30270603551d250420301e06082b0601050507031c
06082b0601050507030106082b06010505070302300e0603551d0f0101fff0404
030201f6300a06082a8648ce3d04030203470030440220744c99008513b2f1bc
fdf9021a46fb174cf883a27cald93faeacf31e4edd12c60220114714dbf51a5e
78f581b9421c6e4702ab537270c5bafb2dl6c3de9aa182c35f
<CODE ENDS>
```

### C.2.3. MASA Certificate

```
<CODE BEGINS>
Certificate:
  Data:
    Version: 3 (0x2)
    Serial Number:
      14:26:b8:1c:ce:d8:c3:e8:14:05:cb:87:67:0d:be:ef:d5:81:25:b4
    Signature Algorithm: ecdsa-with-SHA256
    Issuer: C=NL, ST=NB, L=Helmond, O=vanderstok,
      OU=manufacturer, CN=masa.stok.nl

    Validity
      Not Before: Dec  9 10:02:36 2020 GMT
      Not After : Sep  5 10:02:36 2023 GMT
    Subject: C=NL, ST=NB, L=Helmond, O=vanderstok,
      OU=manufacturer, CN=masa.stok.nl
    Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      Public-Key: (256 bit)
      pub:
        04:59:80:94:66:14:94:20:30:3c:66:08:85:55:86:
```

```
db:e7:d4:d1:d7:7a:d2:a3:1a:0c:73:6b:01:0d:02:
12:15:d6:1f:f3:6e:c8:d4:84:60:43:3b:21:c5:83:
80:1e:fc:e2:37:85:77:97:94:d4:aa:34:b5:b6:c6:
ed:f3:17:5c:f1
ASN1 OID: prime256v1
NIST CURVE: P-256
X509v3 extensions:
  X509v3 Subject Key Identifier:
E4:03:93:B4:C3:D3:F4:2A:80:A4:77:18:F6:96:49:03:01:17:68:A3
  X509v3 Authority Key Identifier:
    keyid:
E4:03:93:B4:C3:D3:F4:2A:80:A4:77:18:F6:96:49:03:01:17:68:A3

  X509v3 Basic Constraints: critical
    CA:TRUE
  X509v3 Extended Key Usage:
    CMC Registration Authority,
    TLS Web Server Authentication,
    TLS Web Client Authentication
  X509v3 Key Usage: critical
    Digital Signature, Non Repudiation, Key Encipherment,
    Data Encipherment, Certificate Sign, CRL Sign
Signature Algorithm: ecdsa-with-SHA256
30:44:02:20:2e:c5:f2:24:72:70:20:ea:6e:74:8b:13:93:67:
8a:e6:fe:fb:8d:56:7f:f5:34:18:a9:ef:a5:0f:c3:99:ca:53:
02:20:3d:dc:91:d0:e9:6a:69:20:01:fb:e4:20:40:de:7c:7d:
98:ed:d8:84:53:61:84:a7:f9:13:06:4c:a9:b2:8f:5c
<CODE ENDS>
```

Below is the hexadecimal representation:

```
<CODE BEGINS>
3082026d30820214a00302010202141426b81cced8c3e81405cb87670dbeefd5
8125b4300a06082a8648ce3d040302306f310b3009060355040613024e4c310b
300906035504080c024e423110300e06035504070c0748656c6d6f6e64311330
11060355040a0c0a76616e64657273746f6b31153013060355040b0c0c6d616e
7566616374757265723115301306035504030c0c6d6173612e73746f6b2e6e6c
301e170d3230313230393130303233365a170d3233303930353130303233365a
306f310b3009060355040613024e4c310b300906035504080c024e423110300e
06035504070c0748656c6d6f6e6431133011060355040a0c0a76616e64657273
746f6b31153013060355040b0c0c6d616e756661637475726572311530130603
5504030c0c6d6173612e73746f6b2e6e6c3059301306072a8648ce3d02010608
2a8648ce3d0301070342000459809466149420303c6608855586dbe7d4d1d77a
d2a31a0c736b010d021215d61ff36ec8d48460433b21c583801efce237857797
94d4aa34b5b6c6edf3175cf1a3818d30818a301d0603551d0e04160414e40393
b4c3d3f42a80a47718f6964903011768a3301f0603551d23041830168014e403
93b4c3d3f42a80a47718f6964903011768a3300f0603551d130101ff04053003
0101ff30270603551d250420301e06082b0601050507031c06082b0601050507
030106082b06010505070302300e0603551d0f0101ff0404030201f6300a0608
2a8648ce3d040302034700304402202ec5f224727020ea6e748b1393678ae6fe
fb8d567ff53418a9efa50fc399ca5302203ddc91d0e96a692001fbe42040de7c
7d98edd884536184a7f913064ca9b28f5c
<CODE ENDS>
```

### C.3. COSE-signed Pledge Voucher Request (PVR)

In this COSE example the voucher request has been signed by the Pledge using the private key of Appendix C.1.1, and has been sent to the link-local JRC (Registrar) over CoAPS.

```
POST coaps://[JRC-link-local-address]/b/rv
Content-Format: TBD3
Payload: signed_request_voucher
```

The payload `signed_request_voucher` is shown as hexadecimal dump (with lf added):

```
<CODE BEGINS>
d28444a101382ea104582097113db094eee8eae48683e7337875c0372164
be89d023a5f3df52699c0fbfb55902d2a11909c5a60274323032302d3132
2d32335431323a30353a32325a0474323032322d31322d32335431323a30
353a32325a01020750684ca83e27230aff97630cf2c1ec409a0d6e706c65
6467652e312e322e332e340a590279308202753082021ca0030201020214
7056eaaa3066d8826a555b9088d462bf9cf28cfd300a06082a8648ce3d04
03023073310b3009060355040613024e4c310b300906035504080c024e42
3110300e06035504070c0748656c6d6f6e6431133011060355040a0c0a76
616e64657273746f6b31143012060355040b0c0b636f6e73756c74616e63
79311a301806035504030c117265676973747261722e73746f6b2e6e6c30
1e170d3230313230393130303233365a170d323131323039313030323336
5a3073310b3009060355040613024e4c310b300906035504080c024e4231
10300e06035504070c0748656c6d6f6e6431133011060355040a0c0a7661
6e64657273746f6b31143012060355040b0c0b636f6e73756c74616e6379
311a301806035504030c117265676973747261722e73746f6b2e6e6c3059
301306072a8648ce3d020106082a8648ce3d03010703420004507ac8491a
8c69c7b5c31d0309ed35ba13f5884ce62b88cf3018154fa059b020ec6beb
b94e02b8934021898da789c711cea71339f50e348edf0d923ed02dc7b7a3
818d30818a301d0603551d0e0416041408c2bf36887f79412185872f16a7
aca6efb3d2b3301f0603551d2304183016801408c2bf36887f7941218587
2f16a7aca6efb3d2b3300f0603551d130101ff040530030101ff30270603
551d250420301e06082b0601050507031c06082b0601050507030106082b
06010505070302300e0603551d0f0101ff0404030201f6300a06082a8648
ce3d04030203470030440220744c99008513b2f1bcfd9021a46fb174cf8
83a27ca1d93faeacf31e4edd12c60220114714dbf51a5e78f581b9421c6e
4702ab537270c5bafb2d16c3de9aa182c35f58473045022063766c7bbd1b
339dbc398e764af3563e93b25a69104befe9aac2b3336b8f56e1022100cd
0419559ad960ccaed4dee3f436eca40b7570b25a52eb60332bc1f2991484
e9
<CODE ENDS>
```

The Pledge uses the "proximity" (SID 2502, enum 2) assertion together with an included proximity-registrar-cert field (SID 2511) to inform MASA about its proximity to the specific Registrar. The representation of signed\_voucher\_request in CBOR diagnostic format is:

```

<CODE BEGINS>
Diagnose(signed_request_voucher) =
18([
h'A101382E',      / {"alg": -47} /
{4: h' 97113DB094EEE8EAE48683E7337875C0372164B
    E89D023A5F3DF52699C0FBFB5'},
h'<request_voucher>', / byte string as detailed below /
h' 3045022063766C7BBD1B339DBC398E764AF3563E93B
25A69104BEFE9AAC2B3336B8F56E1022100CD0419559A
D960CCAED4DEE3F436ECA40B7570B25A52EB60332BC1F
2991484E9'
])

Diagnose(request_voucher) =
{2501: {2: "2020-12-23T12:05:22Z",
         4: "2022-12-23T12:05:22Z",
         1: 2,
         7: h' 684CA83E27230AFF97630CF2C1EC409A',
         13: "pledge.1.2.3.4",
         10: h'<regis-cert-hex>' / byte string as defined in C.2.2 /
        }}
<CODE ENDS>

```

#### C.4. COSE-signed Registrar Voucher Request (RVR)

In this example the Registrar's voucher request has been signed by the JRC (Registrar) using the private key from Appendix C.1.2. Contained within this voucher request is the voucher request PVR that was made by the Pledge to JRC. Note that the RVR uses the HTTPS protocol (not CoAP) and corresponding long URI path names as defined in [RFC8995]. The Content-Type and Accept headers indicate the constrained voucher format that is defined in the present document. Because the Pledge used this format in the PVR, the JRC must also use this format in the RVR.

```

POST https://masa.example.com/.well-known/brski/requestvoucher
Content-Type: application/voucher-cose+cbor
Accept: application/voucher-cose+cbor
Body: signed_masa_request_voucher

```

The payload `signed_masa_voucher_request` is shown as hexadecimal dump (with lf added):

<CODE BEGINS>

```
d28444a101382ea1045820e8735bc4b470c3aa6a7aa9aa8ee584c09c1113
1b205efec5d0313bad84c5cd05590414a11909c5a60274323032302d3132
2d32385431303a30333a33355a0474323032322d31322d32385431303a30
333a33355a07501551631f6e0416bd162ba53ea00c2a050d6e706c656467
652e312e322e332e3405587131322d32385431303a30333a33355a075015
51631f6e0416bd162ba53ea00c2a050d6e706c656467652e312e322e332e
3405587131322d32385431303a3000000000000000000000000000000000416bd16
2ba53ea00c2a050d6e706c656467652e312e322e332e3405587131322d32
385431303a09590349d28444a101382ea104582097113db094eee8eae486
83e7337875c0372164be89d023a5f3df52699c0fbfb55902d2a11909c5a6
0274323032302d31322d32385431303a30333a33355a0474323032322d31
322d32385431303a30333a33355a010207501551631f6e0416bd162ba53e
a00c2a050d6e706c656467652e312e322e332e340a590279308202753082
021ca00302010202147056eaaa3066d8826a555b9088d462bf9cf28cfd30
0a06082a8648ce3d0403023073310b3009060355040613024e4c310b3009
06035504080c024e423110300e06035504070c0748656c6d6f6e64311330
11060355040a0c0a76616e64657273746f6b31143012060355040b0c0b63
6f6e73756c74616e6379311a301806035504030c11726567697374726172
2e73746f6b2e6e6c301e170d3230313230393130303233365a170d323131
3230393130303233365a3073310b3009060355040613024e4c310b300906
035504080c024e423110300e06035504070c0748656c6d6f6e6431133011
060355040a0c0a76616e64657273746f6b31143012060355040b0c0b636f
6e73756c74616e6379311a301806035504030c117265676973747261722e
73746f6b2e6e6c3059301306072a8648ce3d020106082a8648ce3d030107
03420004507ac8491a8c69c7b5c31d0309ed35ba13f5884ce62b88cf3018
154fa059b020ec6bebb94e02b8934021898da789c711cea71339f50e348e
df0d923ed02dc7b7a3818d30818a301d0603551d0e0416041408c2bf3688
7f79412185872f16a7aca6efb3d2b3301f0603551d2304183016801408c2
bf36887f79412185872f16a7aca6efb3d2b3300f0603551d130101ff0405
30030101ff30270603551d250420301e06082b0601050507031c06082b06
01050507030106082b06010505070302300e0603551d0f0101ff04040302
01f6300a06082a8648ce3d04030203470030440220744c99008513b2f1bc
fdf9021a46fb174cf883a27cald93faeacf31e4edd12c60220114714dbf5
1a5e78f581b9421c6e4702ab537270c5bafb2d16c3de9aa182c35f584730
45022063766c7bbdlb339dbc398e764af3563e93b25a69104befe9aac2b3
336b8f56e1022100cd0419559ad960ccaed4dee3f436eca40b7570b25a52
eb60332bc1f2991484e958473045022100e6b45558c1b806bba23f4ac626
c9bdb6fd354ef4330d8dfb7c529f29cca934c802203c1f2ccbbac89733d1
7ee7775bc2654c5f1cc96afba2741cc31532444aa8fed8
```

<CODE ENDS>

The representation of signed\_masa\_voucher\_request in CBOR diagnostic format is:

```

<CODE BEGINS>
Diagnose(signed_registrar_request-voucher)
18([
h'A101382E',      / {"alg": -47} /
h'E8735BC4B470C3AA6A7AA9AA8EE584C09C11131B205EFEC5D0313BAD84
C5CD05'},
h'<registrar_request_voucher>', / byte string as detailed below /
h'3045022100E6B45558C1B806BBA23F4AC626C9BDB6FD354EF4330D8DFB
7C529F29CCA934C802203C1F2CCBBAC89733D17EE7775BC2654C5F1CC96A
FBA2741CC31532444AA8FED8'
])

Diagnose(registrar_request_voucher)
{2501:
  {2: "2020-12-28T10:03:35Z",
    4: "2022-12-28T10:03:35Z",
    7: h'1551631F6E0416BD162BA53EA00C2A05',
    13: "pledge.1.2.3.4",
    5: h'31322D32385431303A30333A333355A07501551631F6E0416BD
      162BA53EA00C2A050D6E706C656467652E312E322E332E3405
      587131322D32385431303A30000000000000000000000000000004
      16BD162BA53EA00C2A050D6E706C656467652E312E322E332E
      3405587131322D32385431303A', / idevid-issuer /
    9: h'<prior-pvr>' / prior-signed-voucher-request = PVR /
  }
}
<CODE ENDS>

```

#### C.5. COSE-signed Voucher from MASA

The resulting voucher is created by the MASA and returned via the JRC to the Pledge. It is signed by the MASA's private key (see Appendix C.1.3) and can be verified by the Pledge using the MASA's public key that it stores.

Below is the binary signed\_voucher, encoded in hexadecimal (with lf added):

```

<CODE BEGINS>
d28444a101382ea104582039920a34ee92d3148ab3a729f58611193270c9
029f7784daf112614b19445d5158cfa1190993a70274323032302d31322d
32335431353a30333a31325a0474323032302d31322d32335431353a3233
3a31325a010007506508e06b2959d5089d7a3169ea889a490b6e706c6564
67652e312e322e332e340858753073310b3009060355040613024e4c310b
300906035504080c024e423110300e06035504070c0748656c6d6f6e6431
133011060355040a0c0a76616e64657273746f6b31143012060355040b0c
0b636f6e73756c74616e6379311a301806035504030c1172656769737472
61722e73746f6b2e6e6c03f458473045022022515d96cd12224ee5d3ac67
3237163bba24ad84815699285d9618f463ee73fa022100a6bff9d8585c1c
9256371ece94da3d26264a5dfec0a354fe7b3aef58344c512f
<CODE ENDS>

```

The representation of signed\_voucher in CBOR diagnostic format is:

```

<CODE BEGINS>
Diagnose(signed_voucher) =
18([
h'A101382E',      / {"alg": -47} /
{4: h'39920A34EE92D3148AB3A729F58611193270C9029F7784DAF112614B194
45D51'},
h'<voucher>',      / byte string as detailed below /
h'3045022022515D96CD12224EE5D3AC673237163BBA24AD84815699285D9618F
463EE73FA022100A6BFF9D8585C1C9256371ECE94DA3D26264A5DFEC0A354FE7B
3AEF58344C512F'
])

Diagnose(voucher) =
{2451:
  {2: "2020-12-23T15:03:12Z",
   4: "2020-12-23T15:23:12Z",
   1: 0,
   7: h'6508E06B2959D5089D7A3169EA889A49',
  11: "pledge.1.2.3.4",
   8: h'<regis-cert-hex>', / as detailed in C.2.2 /
   3: false}
}
<CODE ENDS>

```

In above, regis-cert-hex represents the hexadecimal encoding of the Registrar certificate of Appendix C.2.2.

## Appendix D. Generating Certificates with OpenSSL

This informative appendix shows an example of a Bash shell script to generate test certificates for the Pledge IDevID, the Registrar and the MASA. This shell script cannot be run stand-alone because it depends on particular input files which are not included in this appendix. Nevertheless, this example script may provide guidance on how OpenSSL can be configured for generating Constrained BRSKI certificates.

Note: the \*-comb.crt certificate files combine the certificate with the private key. These are generated to be used by libcoap for DTLS connection establishment.

```
<CODE BEGINS>
#!/bin/bash
#try-cert.sh
export dir=./brski/intermediate
export cadir=./brski
export cnfdir=./conf
export format=pem
export default_crl_days=30
sn=8

DevID=pledge.1.2.3.4
serialNumber="serialNumber=$DevID"
export hwType=1.3.6.1.4.1.6715.10.1
export hwSerialNum=01020304 # Some hex
export subjectAltName="otherName:1.3.6.1.5.5.7.8.4;SEQ:hmodname"
echo $hwType - $hwSerialNum
echo $serialNumber
OPENSSL_BIN="openssl"

# remove all files
rm -r ./brski/*
#
# initialize file structure
# root level
cd $cadir
mkdir certs crl csr newcerts private
chmod 700 private
touch index.txt
touch serial
echo 11223344556600 >serial
echo 1000 >crlnumber
# intermediate level
mkdir intermediate
cd intermediate
```

```
mkdir certs crl csr newcerts private
chmod 700 private
touch index.txt
echo 11223344556600 >serial
echo 1000 > crlnumber
cd ../..

# file structure is cleaned start filling

echo "#####"
echo "create registrar keys and certificates "
echo "#####"

echo "create root registrar certificate using ecDSA with sha 256 key"
$OPENSSL_BIN ecparam -name prime256v1 -genkey \
    -noout -out $cadir/private/ca-regis.key

$OPENSSL_BIN req -new -x509 \
    -config $cnfdir/openssl-regis.cnf \
    -key $cadir/private/ca-regis.key \
    -out $cadir/certs/ca-regis.crt \
    -extensions v3_ca \
    -days 365 \
    -subj "/C=NL/ST=NB/L=Helmond/O=vanderstok/OU=consultancy \
/CN=registrar.stok.nl"

# Combine authority certificate and key
echo "Combine authority certificate and key"
$OPENSSL_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT \
    -inkey $cadir/private/ca-regis.key \
    -in $cadir/certs/ca-regis.crt -export \
    -out $cadir/certs/ca-regis-comb.pfx

# converteer authority pkcs12 file to pem
echo "converteer authority pkcs12 file to pem"
$OPENSSL_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT \
    -in $cadir/certs/ca-regis-comb.pfx \
    -out $cadir/certs/ca-regis-comb.crt -nodes

#show certificate in registrar combined certificate
$OPENSSL_BIN x509 -in $cadir/certs/ca-regis-comb.crt -text

#
# Certificate Authority for MASA
#
```

```
echo "#####"
echo "create MASA keys and certificates "
echo "#####"

echo "create root MASA certificate using ecdsa with sha 256 key"
$OPENSSL_BIN ecparam -name prime256v1 -genkey -noout \
    -out $cadir/private/ca-masa.key

$OPENSSL_BIN req -new -x509 \
    -config $cnfdir/openssl-masa.cnf \
    -days 1000 -key $cadir/private/ca-masa.key \
    -out $cadir/certs/ca-masa.crt \
    -extensions v3_ca \
    -subj "/C=NL/ST=NB/L=Helmond/O=vanderstok/OU=manufacturer\
/CN=masa.stok.nl"

# Combine authority certificate and key
echo "Combine authority certificate and key for masa"
$OPENSSL_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\
    -inkey $cadir/private/ca-masa.key \
    -in $cadir/certs/ca-masa.crt -export \
    -out $cadir/certs/ca-masa-comb.pfx

# converteer authority pkcs12 file to pem for masa
echo "converteer authority pkcs12 file to pem for masa"
$OPENSSL_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\
    -in $cadir/certs/ca-masa-comb.pfx \
    -out $cadir/certs/ca-masa-comb.crt -nodes

#show certificate in pledge combined certificate
$OPENSSL_BIN x509 -in $cadir/certs/ca-masa-comb.crt -text

#
# Certificate for Pledge derived from MASA certificate
#
echo "#####"
echo "create pledge keys and certificates "
echo "#####"

# Pledge derived Certificate

echo "create pledge derived certificate using ecdsa with sha 256 key"
$OPENSSL_BIN ecparam -name prime256v1 -genkey -noout \
    -out $dir/private/pledge.key

echo "create pledge certificate request"
```

```
$OPENSSL_BIN req -nodes -new -sha256 \  
-key $dir/private/pledge.key -out $dir/csr/pledge.csr \  
-subj "/C=NL/ST=NB/L=Helmond/O=vanderstok/OU=manufacturing\  
/CN=uuid:$DevID/$serialNumber"  
  
# Sign pledge derived Certificate  
echo "sign pledge derived certificate "  
$OPENSSL_BIN ca -config $cnfdir/openssl-pledge.cnf \  
-extensions 8021ar_idevid\  
-days 365 -in $dir/csr/pledge.csr \  
-out $dir/certs/pledge.crt  
  
# Add pledge key and pledge certificate to pkcs12 file  
echo "Add derived pledge key and derived pledge \  
certificate to pkcs12 file"  
$OPENSSL_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\  
-inkey $dir/private/pledge.key \  
-in $dir/certs/pledge.crt -export \  
-out $dir/certs/pledge-comb.pfx  
  
# converteer pledge pkcs12 file to pem  
echo "converteer pledge pkcs12 file to pem"  
$OPENSSL_BIN pkcs12 -passin pass:watnietWT -passout pass:watnietWT\  
-in $dir/certs/pledge-comb.pfx \  
-out $dir/certs/pledge-comb.crt -nodes  
  
#show certificate in pledge-comb.crt  
$OPENSSL_BIN x509 -in $dir/certs/pledge-comb.crt -text  
  
#show private key in pledge-comb.crt  
$OPENSSL_BIN ecparam -name prime256v1\  
-in $dir/certs/pledge-comb.crt -text  
  
<CODE ENDS>
```

## Appendix E. Pledge Device Class Profiles

This specification allows implementers to select between various functional options for the Pledge, yielding different code size footprints and different requirements on Pledge hardware. Thus for each product an optimal trade-off between functionality, development/maintenance cost and hardware cost can be made.

This appendix illustrates different selection outcomes by means of defining different example "profiles" of constrained Pledges. In the following subsections, these profiles are defined and a comparison is provided.

## E.1. Minimal Pledge

The Minimal Pledge profile (Min) aims to reduce code size and hardware cost to a minimum. This comes with some severe functional restrictions, in particular:

- \* No support for EST re-enrollment: whenever this would be needed, a factory reset followed by a new bootstrap process is required.
- \* No support for change of Registrar: for this case, a factory reset followed by a new bootstrap process is required.

This profile would be appropriate for single-use devices which must be replaced rather than re-deployed. That might include medical devices, but also sensors used during construction, such as concrete temperature sensors.

## E.2. Typical Pledge

The Typical Pledge profile (Typ) aims to support a typical Constrained BRSKI feature set including EST re-enrollment support and Registrar changes.

## E.3. Full-featured Pledge

The Full-featured Pledge profile (Full) illustrates a Pledge category that supports multiple bootstrap methods, hardware real-time clock, BRSKI/EST resource discovery, and CSR Attributes request/response. It also supports most of the optional features defined in this specification.

## E.4. Comparison Chart of Pledge Classes

The below table specifies the functions implemented in the three example Pledge classes Min, Typ and Full.

Function	Min	Typ	Full
*General*	===	===	====
Support Constrained BRSKI bootstrap	Y	Y	Y
Support other bootstrap method(s)	-	-	Y
Real-time clock and cert time checks	-	-	Y
*Constrained BRSKI*	===	===	====

Discovery for rt=brski*	-	-	Y
Support pinned Registrar public key (RPK)	Y	-	Y
Support pinned Registrar certificate	-	Y	Y
Support pinned Domain CA	-	Y	Y
*Constrained EST*	===	===	=====
Discovery for rt=ace.est*	-	-	Y
GET /att and response parsing	-	-	Y
GET /crtts format 281 (multiple CA certs)	-	-	Y
GET /crtts only format TBD287 (one CA cert only)	Y	Y	-
ETag handling support for GET /crtts	-	Y	Y
Re-enrollment supported	- (1)	Y	Y
6.6.1 optimized procedure	Y	Y	-
Pro-active cert re-enrollment at own initiative	N/A	-	Y
Periodic trust anchor retrieval GET /crtts	- (1)	Y	Y
Supports change of Registrar identity	- (1)	Y	Y

Table 6

Notes: (1) is possible only by doing a factory-reset followed by a new bootstrap procedure.

## Contributors

Russ Housley  
Email: housley@vigilsec.com

Authors' Addresses

Michael Richardson  
Sandelman Software Works  
Email: mcr+ietf@sandelman.ca

Peter van der Stok  
vanderstok consultancy  
Email: stokcons@bbhmail.nl

Panos Kampanakis  
Cisco Systems  
Email: pkampana@cisco.com

Esko Dijk  
IoTconsultancy.nl  
Email: esko.dijk@iotconsultancy.nl

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: January 13, 2022

X. Xiao (Ed.)  
Huawei Technologies  
B. Liu (Ed.)  
A. Hecker  
MRC, Huawei Technologies  
S. Jiang  
Huawei Technologies  
B. Carpenter

School of Computer Science, University of  
Auckland

July 12, 2021

Information Distribution over GRASP  
draft-ietf-anima-grasp-distribution-03

Abstract

Autonomic network infrastructure (ANI) is a generic platform for tenant applications (i.e. AFs). As we will see in some exemplary use cases, AFs may not only require communication capability supported from the infrastructure side, but also the capability the infrastructure can hold and re-distribute information on-demand. This document proposes a set of solutions for information distribution in the ANI. Information distribution is categorized into two different modes: 1) instantaneous distribution and 2) publishing for retrieval. In the former case, the information is sent, propagated and disposed of after reception. In the latter case, information needs to be stored in the network; additionally, conflict resolution is also needed when information stored in the network is updated with proposals from two different AFs.

The capability of information distribution is a fundamental need for an autonomous network ([RFC7575]). This document describes typical use cases of information distribution in ANI and requirements to ANI, such that abundant ways of information distribution can be natively supported. This draft proposes a series of extensions to the autonomic nodes and suggests an implementation based on GRASP ([I-D.ietf-anima-grasp]) extensions as a protocol on the wire.

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 January 13, 2022.

## Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

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

## Table of Contents

1. Introduction . . . . .	3
2. Use Cases of Information Distribution . . . . .	4
2.1. Vehicle-to-Everything (V2X) Communications . . . . .	4
2.2. Service-Based Architecture (SBA) in 3GPP . . . . .	5
2.3. In-Network Computing (INC) . . . . .	7
3. General Requirements of Information Distribution in ANI . . . .	8
4. Node Behaviors . . . . .	10
4.1. Instant Information Distribution (IID) Sub-module . . . . .	10
4.1.1. Instant P2P Communication . . . . .	11
4.1.2. Instant Flooding Communication . . . . .	11
4.2. Asynchronous Information Distribution (AID) Sub-module . .	12
4.2.1. Information Storage . . . . .	12
4.2.2. Event Queue . . . . .	14
4.3. Bulk Information Transfer . . . . .	16
4.4. Summary . . . . .	17
5. Extending GRASP for Information Distribution . . . . .	18
5.1. Realizing Instant P2P Transmission . . . . .	18
5.2. Realizing Instant Selective Flooding . . . . .	18
5.3. Realizing Bulk Information Transfer . . . . .	19
5.4. Realizing Subscription as An Event . . . . .	19
5.5. Un_Subscription Objective Option . . . . .	19

5.6. Publishing Objective Option . . . . .	20
6. Security Considerations . . . . .	20
7. IANA Considerations . . . . .	20
8. Acknowledgements . . . . .	20
9. References . . . . .	21
9.1. Normative References . . . . .	21
9.2. Informative References . . . . .	21
Appendix A. Open Issues [RFC Editor: To Be removed before becoming RFC] . . . . .	22
Appendix B. Closed Issues [RFC Editor: To Be removed before becoming RFC] . . . . .	23
Appendix C. Change log [RFC Editor: To Be removed before becoming RFC] . . . . .	23
Appendix D. Information Distribution Module in ANI . . . . .	23
Appendix E. Asynchronous ID Integrated with GRASP APIs . . . . .	24
Authors' Addresses . . . . .	25

## 1. Introduction

In an autonomic network, autonomic functions (AFs) running on autonomic nodes constantly exchange information, e.g. AF control/management signaling or AF data exchange. This document discusses the information distribution capability of such exchanges among AFs. Many use cases can be abstracted to this model. In the following sections, we will see that the information distribution capability shall become a common denominator in future application scenarios.

In general, depending on the number of participants, the information can be distributed in in the following scenarios:

- 1) Point-to-point (P2P) Communication: information is exchanged between two AFs.
- 2) One-to-Many Communication: information exchanges involve one source AF and multiple receiving AFs.

Approaches of information distribution can be mainly categorized into two basic modes:

- 1) An instantaneous mode (push): a source sends the actual content (e.g. control/management signaling, synchronization data and so on) to all interested receiver(s) immediately. Generally, some preconfigurations are required, where nodes interested in this information must be already known to all nodes because any source AF must be able to decide, to which AFs the data is to be sent.
- 2) An asynchronous mode (delayed pull): here, a source AF publishes the content in some forms in the network, which may later be

looked for, found and retrieved by some endpoints in the AN. Here, depending on the size of the content, either the whole content or only its metadata might be published into the AN. In the latter case the metadata (e.g. a content descriptor, e.g. a key, and a location in the ANI) may be used for the actual retrieval. Importantly, the source, i.e., here as a publisher, needs to be able to determine the location, where the information (or its metadata) can be stored.

Note that in both cases, the total size of transferred information can be larger than the payload size of a single message of a used transport protocol (e.g., Synchronization and Flood messages in GRASP). In this situation, this document also considers a case of bulk data transfer. To avoid repetitive implementations by each AF developer, this document opts for a common support for information distribution implemented as a basic ANI capability. Therefore, it will be available to all AFs. In fact, GRASP already provides part of the capabilities.

Regardless, an AF may still define and implement its own information distribution capability. Such a capability may then be advertised using the common information distribution capability defined in this document. Overall, ANI nodes and AFs may decide, which of the information distribution mechanisms they want to use for which type of information, according to their own preferences.

This document first analyzes requirements for information distribution in autonomic networks (Section 3) and then discuss the relevant node behaviors (Section 4). After that, the required GRASP extensions are formally introduced (Section 5).

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].

## 2. Use Cases of Information Distribution

In this section, we present some important use cases where information distribution is required and ACP's support is commonly needed.

### 2.1. Vehicle-to-Everything (V2X) Communications

The connected Autonomous Driving (AD) vehicles market is driving the evolution of the Internet of Vehicles (IoV) (or Vehicular IoT) and is growing at a five-year compound annual growth rate of 45%, which is 10 times as fast as the overall car market. V2X communication is an inevitable enabling technology that connects vehicles to networks,

where value-added services can be provided and enhance the functionalities of a vehicle. In this section, we introduce some use cases that will be closely relevant to information distribution in an ANI.

- 1) Real-time and High Definition Maps (HDM): In the era of autonomous driving, a digital map not only means for navigation, but real-time and detailed information is required when driving a vehicle. Real-time situational awareness is essential for autonomous vehicles especially at critical road segments in cases of changing road conditions (e.g. new traffic cone detected by another vehicle some time ago). In addition, the relevant high definition local maps have to be available with support from infrastructure side. In this regards, a digital map should not be considered as static information stored on the vehicle, which is spontaneously updated in a periodical manner. Instead, it shall be considered as a dynamic distribution based on information aggregated from the local area and such a distribution shall consider latency requirement. Clearly, the infrastructure side shall be able to hold the information in the network sufficiently close to the relevant area.
- 2) In-car Infotainment: This is another popular use case where in-car data demands will increase significantly in the near future. Today, users their mobile phone to access Internet for retrieving data for work or entertainment purposes. There is already a consensus among OTTs, carriers and car manufacturers that vehicle will become the center of information for passengers onboard. For entertainment, typical scenarios can be stereo HD video streaming and online gaming; for business purposes, examples can be mobile conference. This therefore requires the infrastructure side to be able to schedule and deliver requested information/data to the users with quality-of-service (QoS) considerations.
- 3) Software Update: Software components of connected cars will be remotely maintained in future. Therefore, software update has to be supported by the infrastructure side. Although this can be done by centralized solution where all vehicles access to a central clouds, in terms of load balancing and efficiency, prepared update components can be stored in the network and delivered to endpoints in a distributed manner.

## 2.2. Service-Based Architecture (SBA) in 3GPP

In addition to Internet, carrier networks (i.e. wireless mobile networks) is another world-wide networking system. The current architecture of 5G mobile networks from 3GPP has been defined to follow a service-based architecture (SBA) where any network function

(NF) can be dynamically associated with any other NF(s) when needed to compose a network service. Note that one NF can simultaneously associate with multiple other NFs, instead of being physically wired as in the previous generations of mobile networks. NFs communicate with each other over service-based interface (SBI), which is also standardized by 3GPP [3GPP.23.501].

To realize an SBA network system, detailed requirements are further defined to specify how NFs should interact with each other with information exchange over the SBI. We now list three requirements that are related to information distribution here.

- 1) NF Pub/Sub: Any NF should be able to expose its service status to the network and any NF should be able to subscribe the service status of an NF and get notified if the status is available. A concrete example is that a session management function (SMF) can subscribe to the REGISTER notification from an access management function (AMF) if there is a new user equipment trying to access the mobile network [3GPP.23.502].
- 2) Network Exposure Function (NEF): A particular network function that is required to manage the event exposure and distributions. Specifically, SBA requires such a functionality to register network events from the other NFs (e.g. AMF, SMF and so on), classify the events and properly handle event distributions accordingly in terms of different criteria (e.g. priorities) [3GPP.23.502].
- 3) Network Repository Function (NRF): A particular network function where all service status information is stored for the whole network. An SBA network system requires all NFs to be stateless so as to improve the resilience as well as agility of providing network services. Therefore, the information of the available NFs and the service status generated by those NFs will be globally stored in NRF as a repository of the system. This clearly implies storage capability that keeps the information in the network and provides those information when needed. A concrete example is that whenever a new NF comes up, it first of all registers itself at NRF with its profile. When a network service requires a certain NF, it first inquires NRF to retrieve the availability information and decides whether or not there is an available NF or a new NF must be instantiated [3GPP.23.502].

(Note: 3GPP CT adopted HTTP2.0/JSON to be the protocol communicating between NFs, but autonomic networks can also load HTTP2.0 with in ACP.)

Note that the requirements of the information distribution for ANI control plane are not mentioned here, rather only AF services that have to be necessarily supported by additional information distribution are discussed.

### 2.3. In-Network Computing (INC)

In-network computing recently gets a lot of attentions. INC improves the utilization of the computing resources in the network; INC also brings the processed results closer to the users, which may potentially improves the QoS of network services.

Unlike existing network systems, INC deploys computing tasks directly in the network rather than pushing the tasks to endpoints outside the network. Therefore, a network device is not just a transport device, but a mixture of forwarding, routing and computing. The requires an INC-supported network device having storage by default. Furthermore, computing agents deployed on network nodes will have to communicate with each other by exchanging information. There are several typical applications, where information distribution capability is required, which are summarized below.

- 1) Data Backup: There can be multiple computing agents that are created to serve the same purpose(s). In reality, the multiple agents can run for service resilience, load balancing and so on. This forms a service set. The instances in the service set can be deployed at different locations in the network while they need to keep synchronizing their local states for global consistency. In this case, the computing agents will have to constantly send and receive information across the network.
- 2) Data Aggregation: Multiple computing agents may process different computing tasks but the derived results have to be aggregated or combined. Then a collective result can be derived. In this case, different computing agents collaborate with each other, where information data are exchanged during the processing. A popular example is distributed AI or federated learning applications, where data are stored at different places and model training with the local data is also done in a distributed way. After that, trained models by distributed agents will have to be aggregated. Information distribution will be utilized heavily, combining with local storage.

Clearly, AFs running on network nodes in ANI are the abstraction of the INC use case. AFs can be deployed for both scenarios above.

### 3. General Requirements of Information Distribution in ANI

According to the introduced use cases, the question of information distribution in an autonomic network can be discussed through particular use cases or more generally. Depending on the situation it can be quite simple or might require more complex provisions.

Indeed, in the most general case, the information can be sent:

- 1) at once (in one or multiple packets, in one flow),
- 2) straightaway (send-and-forget),
- 3) to all nodes.

For the first scenario, presuming 1), 2) and 3) hold, information distribution in smaller or scarce topologies can be implemented using broadcast, i.e. unconstrained flooding. For reasons well-understood, this approach has its limits in larger and denser networks. In this case, a graph can be constructed such that it contains every node exactly once (e.g. a spanning tree), still allowing to distribute any information to all nodes straightaway. Multicast tree construction protocols could be used in this case. There are reasonable use cases for such scenarios, as presented in Section 2.

Secondly, a more complex scenario arises, if only 1) and 2) hold, but the information only concerns a subset of nodes. Then, some kinds of selection become required, to which nodes the given information should be distributed. Here, a further distinction is necessary; notably, if the selection of the target nodes is with respect to the nature or position of the node, or whether it is with respect to the information content. If the first, some knowledge about the node types, its topological position, etc (e.g. the routing information within ANI) can be used to distinguish nodes accordingly. For instance, edge nodes and forwarding nodes can be distinguished in this way. If the distribution scope is primarily to be defined by the information elements, then a registration / join / subscription or label distribution mechanism is unavoidable. This would be the case, for instance, if the AFs can be dynamically deployed on nodes, and the information is majorily destined to the AFs. Then, depending on the current AF deployment, the distribution scope must be adjusted as well.

Thirdly, if only 1) holds, but the information content might be required again and again, or might not yet be fully available, then more complex mechanisms might be required to store the information within the network for later, for further redistribution, and for notification of interested nodes. Examples for this include

distribution of reconfiguration information for different AF instances, which might not require an immediate action, but only an eventual update of the parameters. Also, in some situations, there could be a significant delay between the occurrence of a new event and the full content availability (e.g. if the processing requires a lot of time).

Finally, none of the three might hold. Then, along with the subscription and notification, the actual content might be different from its metadata, i.e. some descriptions of the content and, possibly, its location. The fetching can then be implemented in different, appropriate ways, if necessary as a complex transport session.

In essence, as flooding is usually not an option, and the interest of nodes for particular information elements can change over time, ANI should support autonomies also for the information distribution.

This calls for autonomic mechanisms in the ANI, allowing participating nodes to 1) advertise/publish, 2) look for/subscribe to 3) store, 4) fetch/retrieve and 5) instantaneously push data information.

In the following cases, situations depicting complicated ways of information distribution are discussed.

- 1) Long Communication Intervals. The actual sending of the information is not necessarily instantaneous with some events. Sophisticated AFs may involve into longer jobs/tasks (e.g. database lookup, validations, etc.) when processing requests, and might not be able to reply immediately. Instead of actively waiting for the reply, a better way for an interested AF might be to get notified, when the reply is finally available.
- 2) Common Interest Distribution. AFs may share information that is a common interest. For example, the network intent will be distributed to network nodes enrolled, which is usually one-to-many scenario. Intent distribution can also be performed by an instant flooding (e.g. via GRASP) to every network node. However, because of network changes, not every node can be just ready at the moment when the network intent is broadcast. Also, a flooding often does not cover all network nodes as there is usually a limitation on the hop number. In fact, nodes may join in the network sequentially. In this situation, an asynchronous communication model could be a better choice where every (newly joining) node can subscribe the intent information and will get notified if it is ready (or updated).

- 3) Distributed Coordination. With computing and storage resources on autonomic nodes, alive AFs not only consume but also generate data information. An example is AFs coordinating with each other as distributed schedulers, responding to service requests and distributing tasks. It is critical for those AFs to make correct decisions based on local information, which might be asymmetric as well. AFs may also need synthetic/aggregated data information (e.g. statistic info, like average values of several AFs, etc.) to make decisions. In these situations, AFs will need an efficient way to form a global view of the network (e.g. about resource consumption, bandwidth and statistics). Obviously, purely relying on instant communication model is inefficient, while a scalable, common, yet distributed data layer, on which AFs can store and share information in an asynchronous way, should be a better choice.
- 4) Collision Update. Information data not only can be propagated and stored on network nodes in the network, they have to be conflict-free when information is updated especially when there is no central authority available. For example, when two AFs try to propose different updates for the same piece of information that already exist in the network, a decision has to be made for how the existing information shall be updated. Obviously, if this duty has to be handled by individual AFs, the implementation of an AF is too complicated. Therefore, information distribution should consider conflict resolution and provides a set of general solutions for AFs in order to keep information conflict free.

Therefore, for ANI, in order to support various communication scenarios, an information distribution module is required, and both instantaneous and asynchronous communication models should be supported. Some real-world use cases are introduced in Section 2.

#### 4. Node Behaviors

In this section, how a node should behave in order to support the two identified modes of information distribution is discussed. An ANI is a distributed system, so the information distribution module must be implemented in a distributed way as well.

##### 4.1. Instant Information Distribution (IID) Sub-module

In this case, an information sender directly specifies the information receiver(s). The instant information distribution sub-module will be the main element.

#### 4.1.1. Instant P2P Communication

IID sub-module performs instant information transmission for ASAs running in an ANI. In specific, IID sub-module will have to retrieve the address of the information receiver specified by an ASA, then deliver the information to the receiver. Such a delivery can be done either in a connectionless or a connection-oriented way.

Current GRASP provides the capability to support instant P2P synchronization for ASAs. A P2P synchronization is a use case of P2P information transmission. However, as mentioned in Section 3, there are some scenarios where one node needs to transmit some information to another node(s). This is different to synchronization because after transmitting the information, the local status of the information does not have to be the same as the information sent to the receiver. This is not directly support by existing GRASP.

#### 4.1.2. Instant Flooding Communication

IID sub-module finishes instant flooding for ASAs in an ANI. Instant flooding is for all ASAs in an ANI. An information sender has to specify a special destination address of the information and broadcast to all interfaces to its neighbors. When another IID sub-module receives such a broadcast, after checking its TTL, it further broadcast the message to the neighbors. In order to avoid flooding storms in an ANI, usually a TTL number is specified, so that after a pre-defined limit, the flooding message will not be further broadcast again.

In order to avoid unnecessary flooding, a selective flooding can be done where an information sender wants to send information to multiple receivers at once. When doing this, sending information needs to contain criteria to judge on which interfaces the distributed information should and should not be sent. Specifically, the criteria contain:

- o Matching Condition: a set of matching rules such as addresses of recipients, node features and so on.
- o Action: what the node needs to do when the Matching Condition is fulfilled. For example, the action could be forwarding or discarding the distributed message.

Sent information must be included in the message distributed from the sender. The receiving node reacts by first checking the carried Matching Condition in the message to decide who should consume the message, which could be either the node itself, some neighbors or

both. If the node itself is a recipient, Action field is followed; if a neighbor is a recipient, the message is sent accordingly.

An exemplary extension to support selective flooding on GRASP is described in Section 5.

#### 4.2. Asynchronous Information Distribution (AID) Sub-module

In asynchronous information distribution, sender(s) and receiver(s) are not immediately specified while they may appear in an asynchronous way. Firstly, AID sub-module enables that the information can be stored in the network; secondly, AID sub-module provides an information publication and subscription (Pub/Sub) mechanism for ASAs.

As sketched in the previous section, in general each node requires two modules: 1) Information Storage (IS) module and 2) Event Queue (EQ) module in the information distribution module. Details of the two modules are described in the following sections.

##### 4.2.1. Information Storage

IS module handles how to save and retrieve information for ASAs across the network. The IS module uses a syntax to index information, generating the hash index value (e.g. a hash value) of the information and mapping the hash index to a certain node in ANI. Note that, this mechanism can use existing solutions. Specifically, storing information in an ANIMA network will be realized in the following steps.

- 1) ASA-to-IS Negotiation. An ASA calls the API provided by information distribution module (directly supported by IS sub-module) to request to store the information somewhere in the network. The IS module performs various checks of the request (e.g. permitted information size).
- 2) Storing Peer Mapping. The information block will be handled by the IS module in order to calculate/map to a peer node in the network. Since ANIMA network is a peer-to-peer network, a typical way is to use distributed hash table (DHT) to map information to a unique index identifier. For example, if the size of the information is reasonable, the information block itself can be hashed, otherwise, some meta-data of the information block can be used to generate the mapping.
- 3) Storing Peer Negotiation Request. Negotiation request of storing the information will be sent from the IS module to the IS module on the destination node. The negotiation request contains

parameters about the information block from the source IS module. According to the parameters as well as the local available resource, the requested storing peer will send feedback the source IS module.

- 4) Storing Peer Negotiation Response. Negotiation response from the storing peer is sent back to the source IS module. If the source IS module gets confirmation that the information can be stored, source IS module will prepare to transfer the information block; otherwise, a new storing peer must be discovered (i.e. going to step 7).
- 5) Information Block Transfer. Before sending the information block to the storing peer that already accepts the request, the IS module of the source node will check if the information block can be afforded by one GRASP message. If so, the information block will be directly sent by calling a GRASP API ([I-D.ietf-anima-grasp-api]). Otherwise, a bulk data transmission is needed. For that, there are multiple ways to do it. The first option is to utilize one of existing protocols that is independent of the GRASP stack. For example, a session connectivity can be established to the storing peer, and over the connection the bulky data can be transmitted part by part. In this case, the IS module should support basic TCP-based session protocols such as HTTP(s) or native TCP. The second option is to directly use GRASP itself for bulky data transferring [I-D.carpenter-anima-grasp-bulk].
- 6) Information Writing. Once the information block (or a smaller block) is received, the IS module of the storing peer will store the data block in the local storage is accessible.
- 7) (Optional) New Storing Peer Discovery. If the previously selected storing peer is not available to store the information block, the source IS module will have to identify a new destination node to start a new negotiation. In this case, the discovery can be done by using discovery GRASP API to identify a new candidate, or more complex mechanisms can be introduced.

Similarly, Getting information from an ANI will be realized in the following steps.

- 1) ASA-to-IS Request. An ASA accesses the IS module via the APIs exposed by the information distribution module. The key/index of the interested information will be sent to the IS module. An assumption here is that the key/index should be known to an ASA before an ASA can ask for the information. This relates to the publishing/subscribing of the information, which are handled by other modules (e.g. Event Queue with Pub/Sub supported by GRASP).

- 2) Storing Peer Mapping. IS module maps the key/index of the requested information to a peer that stores the information, and prepares the information request. The mapping here follows the same mechanism when the information is stored.
- 3) Retrieval Negotiation Request. The source IS module sends a request to the storing peer and asks if such an information object is available.
- 4) Retrieval Negotiation Response. The storing peer checks the key/index of the information in the request, and replies to the source IS module. If the information is found and the information block can be afforded within one GRASP message, the information will be sent together with the response to the source IS module.
- 5) (Optional) New Destination Request. If the information is not found after the source IS module gets the response from the originally identified storing peer, the source IS module will have to discover the location of the requested information.

IS module can reuse distributed databases and key value stores like NoSQL, Cassandra, DHT technologies. storage and retrieval of information are all event-driven responsible by the EQ module.

#### 4.2.2. Event Queue

The Event Queue (EQ) module is to help ASAs to publish information to the network and subscribe to interested information in asynchronous scenarios. In an ANI, information generated on network nodes is an event labeled with an event ID, which is semantically related to the topic of the information. Key features of EQ module are summarized as follows.

- 1) Event Group: An EQ module provides isolated queues for different event groups. If two groups of AFs could have completely different purposes, the EQ module allows to create multiple queues where only AFs interested in the same topic will be aware of the corresponding event queue.
- 2) Event Prioritization: Events can have different priorities in ANI. This corresponds to how much important or urgent the event implies. Some of them are more urgent than regular ones. Prioritization allows AFs to differentiate events (i.e. information) they publish or subscribe to.
- 3) Event Matching: an information consumer has to be identified from the queue in order to deliver the information from the provider. Event matching keeps looking for the subscriptions in the queue to

see if there is an exact published event there. Whenever a match is found, it will notify the upper layer to inform the corresponding ASAs who are the information provider and subscriber(s) respectively.

The EQ module on every network node operates as follows.

- 1) Event ID Generation: If information of an ASA is ready, an event ID is generated according to the content of the information. This is also related to how the information is stored/saved by the IS module introduced before. Meanwhile, the type of the event is also specified where it can be of control purpose or user plane data.
- 2) Priority Specification: According to the type of the event, the ASA may specify its priority to say how this event is to be processed. By considering both aspects, the priority of the event will be determined.
- 3) Event Enqueue: Given the event ID, event group and its priority, a queue is identified locally if all criteria can be satisfied. If there is such a queue, the event will be simply added into the queue, otherwise a new queue will be created to accommodate such an event.
- 4) Event Propagation: The published event will be propagated to the other network nodes in the ANIMA domain. A propagation algorithm can be employed to optimize the propagation efficiency of the updated event queue states.
- 5) Event Match and Notification: While propagating updated event states, EQ module in parallel keeps matching published events and its interested consumers. Once a match is found, the provider and subscriber(s) will be notified for final information retrieval.

The category of event priority is defined as the following. In general, there are two event types:

- 1) Network Control Event: This type of events are defined by the ANI for operational purposes on network control. A pre-defined priority levels for required system messages is suggested. For highest level to lowest level, the priority value ranges from NC\_PRIOR\_HIGH to NC\_PRIOR\_LOW as integer values. The NC\_PRIOR\_\* values will be defined later according to the total number system events required by the ANI.
- 2) Custom ASA Event: This type of events are defined by the ASAs of users. This specifies the priority of the message within a group

of ASAs, therefore it is only effective among ASAs that join the same message group. Within the message group, a group header/leader has to define a list of priority levels ranging from CUST\_PRIOR\_HIGH to CUST\_PRIOR\_LOW. Such a definition completely depends on the individual purposes of the message group. When a system message is delivered, its event type and event priority value have to be both specified.

Event contains the address where the information is stored, after a subscriber is notified, it directly retrieves the information from the given location.

#### 4.3. Bulk Information Transfer

In both cases discussed previously, they are limited to distributing GRASP Objective Options contained in messages that cannot exceed the GRASP maximum message size of 2048 bytes. This places a limit on the size of data that can be transferred directly in a GRASP message such as a Synchronization or Flood operation for instantaneous information distribution.

There are scenarios in autonomic networks where this restriction is a problem. One example is the distribution of network policy in lengthy formats such as YANG or JSON. Another case might be an Autonomic Service Agent (ASA) uploading a log file to the Network Operations Center (NOC). A third case might be a supervisory system downloading a software upgrade to an autonomic node. A related case might be installing the code of a new or updated ASA to a target node.

Naturally, an existing solution such as a secure file transfer protocol or secure HTTP might be used for this. Other management protocols such as syslog [RFC5424] or NETCONF [RFC6241] might also be used for related purposes, or might be mapped directly over GRASP. The present document, however, applies to any scenario where it is preferable to re-use the autonomic networking infrastructure itself to transfer a significant amount of data, rather than install and configure an additional mechanism.

The node behavior is to use the GRASP Negotiation process to transfer and acknowledge multiple blocks of data in successive negotiation steps, thereby overcoming the GRASP message size limitation. The emphasis is placed on simplicity rather than efficiency, high throughput, or advanced functionality. For example, if a transfer gets out of step or data packets are lost, the strategy is to abort the transfer and try again. In an enterprise network with low bit error rates, and with GRASP running over TCP, this is not considered a serious issue. Clearly, a more sophisticated approach could be

designed but if the application requires that, existing protocols could be used, as indicated in the preceding paragraph.

As for any GRASP operation, the two participants are considered to be Autonomic Service Agents (ASAs) and they communicate using a specific GRASP Objective Option, containing its own name, some flag bits, a loop count, and a value. In bulk transfer, we can model the ASA acting as the source of the transfer as a download server, and the destination as a download client. No changes or extensions are required to GRASP itself, but compared to a normal GRASP negotiation, the communication pattern is slightly asymmetric:

- 1) The client first discovers the server by the GRASP discovery mechanism (M\_DISCOVERY and M\_RESPONSE messages).
- 2) The client then sends a GRASP negotiation request (M\_REQ\_NEG message). The value of the objective expresses the requested item (e.g., a file name - see the next section for a detailed example).
- 3) The server replies with a negotiation step (M\_NEGOTIATE message). The value of the objective is the first section of the requested item (e.g., the first block of the requested file as a raw byte string).
- 4) The client replies with a negotiation step (M\_NEGOTIATE message). The value of the objective is a simple acknowledgement (e.g., the text string 'ACK').

The last two steps repeat until the transfer is complete. The server signals the end by transferring an empty byte string as the final value. In this case the client responds with a normal end to the negotiation (M\_END message with an O\_ACCEPT option).

Errors of any kind are handled with the normal GRASP mechanisms, in particular by an M\_END message with an O\_DECLINE option in either direction. In this case the GRASP session terminates. It is then the client's choice whether to retry the operation from the start, as a new GRASP session, or to abandon the transfer. The block size must be chosen such that each step does not exceed the GRASP message size limit of 2048 bits.

#### 4.4. Summary

In summary, the general requirements for the information distribution module on each autonomic node are realized by two sub-modules handling instant communications and asynchronous communications, respectively. For instantaneous mode, node requirements are simple,

calling for support for additional signaling. With minimum efforts, reusing the existing GRASP is possible.

For asynchronous mode, information distribution module uses new primitives on the wire, and implements an event queue and an information storage mechanism. An architectural consideration on ANI with the information distribution module is briefly discussed in Appendix D.

In both cases, a scenario of bulk information transfer is considered where the retrieved information cannot be fitted in one GRASP message. Based on GRASP Negotiation operation, multiple transmissions can be repeatedly done in order to transfer bulk information piece by piece.

## 5. Extending GRASP for Information Distribution

### 5.1. Realizing Instant P2P Transmission

This could be a new message in GRASP. In fragmentary CDDL, an Un-solicited Synchronization message follows the pattern:

```
unsolicited_synch-message = [M_UNSOLIDSYNCH, session-id,  
objective]
```

A node MAY actively send a unicast Un-solicited Synchronization message with the Synchronization data, to another node. This MAY be sent to port GRASP\_LISTEN\_PORT at the destination address, which might be obtained by GRASP Discovery or other possible ways. The synchronization data are in the form of GRASP Option(s) for specific synchronization objective(s).

### 5.2. Realizing Instant Selective Flooding

Since normal flooding is already supported by GRASP, this section only defines the selective flooding extension.

In fragmentary CDDL, the selective flooding follows the pattern:

```
selective-flood-option = [O_SELECTIVE_FLOOD, +O_MATCH-CONDITION,  
match-object, action]  
  
O_MATCH-CONDITION = [O_MATCH-CONDITION, Obj1, match-rule, Obj2]  
Obj1 = text  
  
match-rule = GREATER / LESS / WITHIN / CONTAIN  
  
Obj2 = text
```

```
match-object = NEIGHBOR / SELF
```

```
action = FORWARD / DROP
```

The option field encapsulates a match-condition option which represents the conditions regarding to continue or discontinue flood the current message. For the match-condition option, the Obj1 and Obj2 are to objects that need to be compared. For example, the Obj1 could be the role of the device and Obj2 could be "RSG". The match rules between the two objects could be greater, less than, within, or contain. The match-object represents of which Obj1 belongs to, it could be the device itself or the neighbor(s) intended to be flooded. The action means, when the match rule applies, the current device just continues flood or discontinues.

### 5.3. Realizing Bulk Information Transfer

### 5.4. Realizing Subscription as An Event

In fragmentary CDDL, a Subscription Objective Option follows the pattern:

```
subscription-objection-option = [SUBSCRIPTION, 2, 2, subobj]
objective-name = SUBSCRIPTION

objective-flags = 2

loop-count = 2

subobj = text
```

This option MAY be included in GRASP M\_Synchronization, when included, it means this message is for a subscription to a specific object.

### 5.5. Un\_Subscription Objective Option

In fragmentary CDDL, a Un\_Subscribe Objective Option follows the pattern:

```
Unsubscribe-objection-option = [UNSUBSCRIB, 2, 2, unsubobj]

objective-name = SUBSCRIPTION

objective-flags = 2

loop-count = 2
```

unsubobj = text

This option MAY be included in GRASP M\_Synchronization, when included, it means this message is for a un-subscription to a specific object.

#### 5.6. Publishing Objective Option

In fragmentary CDDL, a Publish Objective Option follows the pattern:

publish-objection-option = [PUBLISH, 2, 2, pubobj]

objective-name = PUBLISH

objective-flags = 2

loop-count = 2

pubobj = text

This option MAY be included in GRASP M\_Synchronization, when included, it means this message is for a publish of a specific object data.

#### 6. Security Considerations

The distribution source authentication could be done at multiple layers:

- o Outer layer authentication: the GRASP communication is within ACP ([I-D.ietf-anima-autonomic-control-plane]). This is the default GRASP behavior.
- o Inner layer authentication: the GRASP communication might not be within a protected channel, then there should be embedded protection in distribution information itself. Public key infrastructure might be involved in this case.

#### 7. IANA Considerations

TBD.

#### 8. Acknowledgements

Valuable comments were received from Zoran Despotovic, Brian Carpenter, Michael Richardson, Roland Bless, Mohamed Boucadair, Diego Lopez, Toerless Eckert and other participants in the ANIMA working group.

This document was produced using the xml2rfc tool [RFC2629].

## 9. References

### 9.1. Normative References

- [I-D.ietf-anima-grasp]  
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-15 (work in progress), July 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <<https://www.rfc-editor.org/info/rfc2629>>.

### 9.2. Informative References

- [I-D.carpenter-anima-grasp-bulk]  
Carpenter, B., Jiang, S., and B. Liu, "Transferring Bulk Data over the GeneRIC Autonomic Signaling Protocol (GRASP)", draft-carpenter-anima-grasp-bulk-05 (work in progress), January 2020.
- [I-D.du-anima-an-intent]  
Du, Z., Jiang, S., Nobre, J. C., Ciavaglia, L., and M. Behringer, "ANIMA Intent Policy and Format", draft-du-anima-an-intent-05 (work in progress), February 2017.
- [I-D.ietf-anima-autonomic-control-plane]  
Eckert, T., Behringer, M. H., and S. Bjarnason, "An Autonomic Control Plane (ACP)", draft-ietf-anima-autonomic-control-plane-30 (work in progress), October 2020.
- [I-D.ietf-anima-bootstrapping-keyinfra]  
Pritikin, M., Richardson, M. C., Eckert, T., Behringer, M. H., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-45 (work in progress), November 2020.

[I-D.ietf-anima-grasp-api]

Carpenter, B., Liu, B., Wang, W., and X. Gong, "Generic Autonomic Signaling Protocol Application Program Interface (GRASP API)", draft-ietf-anima-grasp-api-10 (work in progress), January 2021.

[I-D.ietf-anima-reference-model]

Behringer, M. H., Carpenter, B., Eckert, T., Ciavaglia, L., and J. C. Nobre, "A Reference Model for Autonomic Networking", draft-ietf-anima-reference-model-10 (work in progress), November 2018.

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <<https://www.rfc-editor.org/info/rfc7575>>.

#### Appendix A. Open Issues [RFC Editor: To Be removed before becoming RFC]

1. More reference to the use cases in the introduction.
2. Better explanation of the required context of the Connected-Car case: Not applicable unless the ACP will be extended to the car, which may not be desirable with the current ACP design, but maybe refocussing on an "autonomous fleet" use-case (e.g.: all cars operated by some taxi like service).
3. Consider use-case/example of firmware update. By abstracting the location of the firmware from the name of the firmware, while providing a way to notify about it, this significantly supports distribution of firmware updates. References to SUIT would be appropriate.
4. Issues discussed in [https://mailarchive.ietf.org/arch/msg/anima/\\_0fYQPBcLPt8xzQee7P4dILsn3A](https://mailarchive.ietf.org/arch/msg/anima/_0fYQPBcLPt8xzQee7P4dILsn3A)
5. Rethink/refine terminology, e.g.: "module" seems to be too prescriptive. Refine proposed extensions.
6. Provide more protocol behavior description instead of only implementation / software module architecture description. Reduce the latter or provide better justification for their presence due to explained interoperability requirements.
7. Better motivation in sections 1..4 of the proposed extensions

8. Consider moving examples from appendices into core-text . Ideally craft a single use-case showing/applying all extensions (most simple use case that uses them all).
9. Refine terminology to better match/reuse-the established terminology from the pre-existing ANIMA documents.

Appendix B. Closed Issues [RFC Editor: To Be removed before becoming RFC]

Appendix C. Change log [RFC Editor: To Be removed before becoming RFC]

draft-ietf-anima-grasp-distribution-00, 2020-02-25:

File name changed following WG adoption.

\_\_Added appendix A&B for open/closed issues. The open issues were comments received during the adoption call.

Appendix D. Information Distribution Module in ANI

This appendix describes how the information distribution module fits into the ANI and what extensions of GRASP are required.

(preamble)

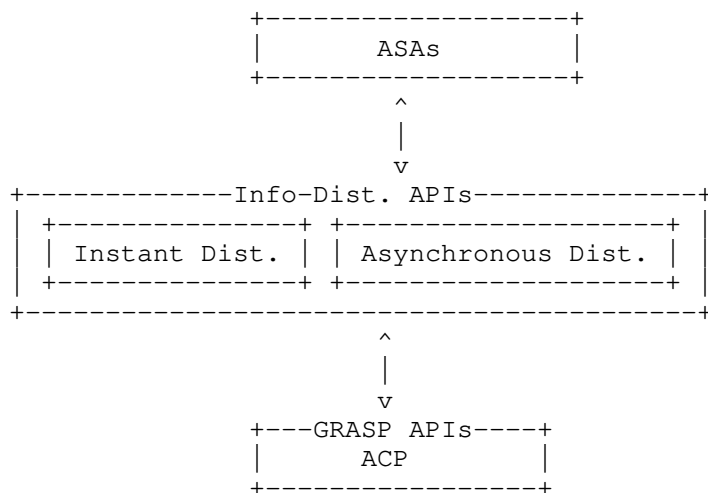


Figure E.1 Information Distribution Module and GRASP Extension.

As the Fig 1 shows, the information distribution module two sub-modules for instant and asynchronous information distributions,

respectively, and provides APIs to ASAs. Specific Behaviors of modules are described in Section 5.

#### Appendix E. Asynchronous ID Integrated with GRASP APIs

Actions triggered to the information distribution module will eventually invoke underlying GRASP APIs. Moreover, EQ and IS modules are usually correlated. When an AF(ASA) publishes information, not only such an event is translated and sent to EQ module, but also the information is indexed and stored simultaneously. Similarly, when an AF(ASA) subscribes information, not only subscribing event is triggered and sent to EQ module, but also the information will be retrieved by IS module at the same time.

- o Storing and publishing information: This action involves both IS and EQ modules where a node that can store the information will be discovered first and related event will be published to the network. For this, GRASP APIs `discover()`, `synchronize()` and `flood()` are combined to compose such a procedure. In specific, `discover()` call will specify its objective being to "store\_data" and the return parameters could be either an `ASA_locator` who will accept to store the data, or an error code indicating that no one could afford such data; after that, `synchronize()` call will send the data to the specified `ASA_locator` and the data will be stored at that node, with return of processing results like `store_data_ack`; meanwhile, such a successful event (i.e. data is stored successfully) will be flooded via a `flood()` call to interesting parties (such a multicast group existed).
- o Subscribing and getting information: This action involves both IS and EQ modules as well where a node that is interested in a topic will subscribe the topic by triggering EQ module and if the topic is ready IS module will retrieve the content of the topic (i.e. the data). GRASP APIs such as `register_objective()`, `flood()`, `synchronize()` are combined to compose the procedure. In specific, any subscription action received by EQ module will be translated to `register_objective()` call where the interested topic will be the parameter inside of the call; the registration will be (selectively) flooded to the network by an API call of `flood()` with the option we extended in this draft; once a matched topic is found (because of the previous procedure), the node finding such a match will call API `synchronize()` to send the stored data to the subscriber.

## Authors' Addresses

Xun Xiao  
Huawei Technologies  
Q5, Huawei Campus  
No.156 Beiqing Road  
Hai-Dian District, Beijing 100095  
P.R. China

Email: leo.liubing@huawei.com

Bing Liu  
MRC, Huawei Technologies  
German Research Center  
Huawei Technologies  
Riesstr. 25  
Muenchen 80992  
Germany

Email: xun.xiao@huawei.com

Artur Hecker  
MRC, Huawei Technologies  
German Research Center  
Huawei Technologies  
Riesstr. 25  
Muenchen 80992  
Germany

Email: artur.hecker@huawei.com

Sheng Jiang  
Huawei Technologies  
Q27, Huawei Campus  
No.156 Beiqing Road  
Hai-Dian District, Beijing 100095  
P.R. China

Email: jiangsheng@huawei.com

Brian  
School of Computer Science, University of  
Auckland  
PB 92019  
Auckland 1142  
New Zealand

Email: [brian.e.carpenter@gmail.com](mailto:brian.e.carpenter@gmail.com)

anima Working Group  
Internet-Draft  
Updates: RFC8366 (if approved)  
Intended status: Standards Track  
Expires: 8 September 2022

M. Richardson  
Sandelman Software Works  
T. Werner  
Siemens AG  
7 March 2022

JWS signed Voucher Artifacts for Bootstrapping Protocols  
draft-ietf-anima-jws-voucher-03

## Abstract

RFC8366 defines a digital artifact called voucher as a YANG-defined JSON document that has been signed using a Cryptographic Message Syntax (CMS) structure. This memo introduces a variant of the voucher structure in which CMS is replaced by the JSON Object Signing and Encryption (JOSE) mechanism described in RFC7515 to better support use-cases in which JOSE is preferred over CMS.

In addition to explaining how the format is created, MIME types are registered and examples are provided.

## 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 8 September 2022.

## Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights

and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. JSON Web Signatures - General JWS JSON Serialization	
Syntax . . . . .	3
3.1. Unprotected Header . . . . .	4
3.2. Protected Header . . . . .	4
3.3. Voucher Representation in General JWS JSON Serialization	
Syntax . . . . .	4
4. Privacy Considerations . . . . .	5
5. Security Considerations . . . . .	5
6. IANA Considerations . . . . .	6
6.1. Media-Type Registry . . . . .	6
6.1.1. application/voucher-jws+json . . . . .	6
7. Changelog . . . . .	6
8. References . . . . .	7
8.1. Normative References . . . . .	7
8.2. Informative References . . . . .	7
Appendix A. Examples . . . . .	9
A.1. Example Pledge Voucher Request - PVR (from Pledge to Registrar) . . . . .	9
A.2. Example Parboiled Registrar Voucher Request - RVR (from Registrar to MASA) . . . . .	10
A.3. Example Voucher Response (from MASA to Pledge, via Registrar) . . . . .	12
Authors' Addresses . . . . .	14

## 1. Introduction

"A Voucher Artifact for Bootstrapping Protocols", [RFC8366] describes a voucher artifact used in "Bootstrapping Remote Secure Key Infrastructure" [BRSKI] and "Secure Zero Touch Provisioning" [SZTP] to transfer ownership of a device from a manufacturer to an owner. That document defines the base YANG module, and also the initial serialization to JSON [RFC8259], with a signature provided by [RFC5652].

Other work, [I-D.ietf-anima-constrained-voucher] provides a mapping of the YANG to CBOR [RFC8949] with a signature format of COSE [RFC8812].

This document provides an equivalent mapping of JSON format with the signature format as JSON Web Signature (JWS) [RFC7515]. The encoding specified in this document is required for [I-D.ietf-anima-brski-prm] and may be required and/or preferred in other use cases, for example when JWS is already used in other parts of the use case, but CMS is not.

This document does not extend the YANG definition of [RFC8366] at all, but accepts that other efforts such as [I-D.richardson-anima-voucher-delegation], [I-D.friel-anima-brski-cloud], and [I-D.ietf-anima-brski-prm] do. This document supports signing any of the extended schemas defined in those documents and any new documents that may appear after this one.

With the availability of different encoded vouchers, it is up to an industry specific application statement to indicate/decide which voucher signature format is to be used. There is no provision across the different voucher signature formats that a receiver could safely recognize which format it uses unless additional context is provided. For example, [BRSKI] provides this context via the MIME-Type for the voucher payload.

This document should be considered an Update to [RFC8366] in the category of "See Also" as per [I-D.kuehlewind-update-tag].

## 2. Terminology

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

## 3. JSON Web Signatures - General JWS JSON Serialization Syntax

[RFC Editor: please delete] /\* TODO: ... \*/

[RFC7515] defines two serializations: the "JWS Compact Serialization" and the "JWS JSON Serialization".

The [RFC8366] JSON structure consists of a nested map, the outer part of which is:

```
{ "ietf-voucher:voucher" : { some inner items }}
```

this is considered the JSON payload as described in [RFC7515] section 3.

A JWS JSON Serialization Overview is given by [RFC7515] in section 3.2 and section 7.2.1 provides more details. It works out to:

```
[RFC Editor: please delete] /*  
TODO: ...  
*/
```

There are a number of attributes. They are:

### 3.1. Unprotected Header

```
[RFC Editor: please delete] /* TODO: ... */
```

### 3.2. Protected Header

The standard "typ" and "alg" values described in [RFC7515] are expected in the protected headers.

It remains to be determined (XXX), what values, if any, should go into the "typ" header, as in the [BRSKI] use cases, there are additional HTTP MIME type headers to indicate content types.

The "alg" should contain the algorithm type such as "ES256".

If PKIX [RFC5280] format certificates are used then the [RFC7515] section 4.1.6 "x5c" certificate chain SHOULD be used to contain the certificate and chain. Vouchers will often need all certificates in the chain, including what would be considered the trust anchor certificate because intermediate devices (such as the Registrar) may need to audit the artifact, or end systems may need to pin a trust anchor for future operations. This is consistent with [BRSKI] section 5.5.2.

### 3.3. Voucher Representation in General JWS JSON Serialization Syntax

```
{
  "payload": {
    "ietf-voucher:voucher": {
      "assertion": "logged",
      "serial-number": "0123456789",
      "nonce": "5742698422680472",
      "created-on": "2022-03-02T03:01:24.618Z",
      "pinned-domain-cert": "base64encodedvalue=="
    }
  },
  "signatures": [
    {
      "protected": {
        "x5c": [
          "base64encodedvalue=="
        ],
        "alg": "ES256"
      },
      "signature": "base64encodedvalue=="
    }
  ]
}
```

Figure 1: Voucher Representation in General JWS JSON  
Serialization Syntax

#### 4. Privacy Considerations

The Voucher Request reveals the IDevID of the component (Pledge) that is on-boarding.

This request occurs over HTTP-over-TLS, however the Pledge to Registrar transaction is over a provisional TLS session, and it is subject to disclosure via by a Dolev-Yao attacker (a "malicious messenger") [onpath]. This is explained in [BRSKI] section 10.2.

The use of a JWS header brings no new privacy considerations.

#### 5. Security Considerations

The issues of how [RFC8366] vouchers are used in a [BRSKI] system is addressed in section 11 of that document. This document does not change any of those issues, it just changes the signature technology used for vouchers and voucher requests.

[SZTP] section 9 deals with voucher use in Secure Zero Touch Provisioning, and this document also makes no changes to security.

## 6. IANA Considerations

### 6.1. Media-Type Registry

This section registers the 'application/voucher-jws+json' in the "Media Types" registry.

#### 6.1.1. application/voucher-jws+json

Type name: application  
Subtype name: voucher-jws+json  
Required parameters: none  
Optional parameters: none  
Encoding considerations: JWS+JSON vouchers are JOSE objects signed with one signer.  
Security considerations: See Security Considerations, Section  
Interoperability considerations: The format is designed to be broadly interoperable.  
Published specification: THIS RFC.  
Applications that use this media type: ANIMA, 6tisch, and other zero-touch imprinting systems  
Additional information:  
  Magic number(s): None  
  File extension(s): .vjj  
  Macintosh file type code(s): none  
Person & email address to contact for further information: IETF ANIMA WG  
Intended usage: LIMITED  
Restrictions on usage: NONE  
Author: ANIMA WG  
Change controller: IETF  
Provisional registration? (standards tree only): NO

## 7. Changelog

- \* Added adoption call comments from Toerless. Changed from [RFCxxxx] to [THING] style for some key references.
- \* Updated references "I-D.ietf-anima-brski-async-enroll" switched to "I-D.ietf-anima-brski-prm"
- \* Switch from "JWS Compact Serialization" to "General JWS JSON Serialization", as focus is now on "General JWS JSON Serialization"
- \* Include Voucher representation in "General JWS JSON Serialization" syntax

- \* Include examples A1, A2, A3 using "General JWS JSON Serialization"

## 8. References

### 8.1. Normative References

- [BRSKI] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <<https://www.rfc-editor.org/info/rfc8995>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <<https://www.rfc-editor.org/info/rfc7515>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", RFC 8366, DOI 10.17487/RFC8366, May 2018, <<https://www.rfc-editor.org/info/rfc8366>>.
- [SZTP] Watsen, K., Farrer, I., and M. Abrahamsson, "Secure Zero Touch Provisioning (SZTP)", RFC 8572, DOI 10.17487/RFC8572, April 2019, <<https://www.rfc-editor.org/info/rfc8572>>.

### 8.2. Informative References

- [I-D.friel-anima-brski-cloud] Friel, O., Shekh-Yusef, R., and M. Richardson, "BRSKI Cloud Registrar", Work in Progress, Internet-Draft, draft-friel-anima-brski-cloud-04, 6 April 2021, <<https://www.ietf.org/archive/id/draft-friel-anima-brski-cloud-04.txt>>.

- [I-D.ietf-anima-brski-prm]  
Fries, S., Werner, T., Lear, E., and M. C. Richardson,  
"BRSKI with Pledge in Responder Mode (BRSKI-PRM)", Work in Progress, Internet-Draft, draft-ietf-anima-brski-prm-02, 4 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-anima-brski-prm-02.txt>>.
- [I-D.ietf-anima-constrained-voucher]  
Richardson, M., Stok, P. V. D., Kampanakis, P., and E. Dijk, "Constrained Bootstrapping Remote Secure Key Infrastructure (BRSKI)", Work in Progress, Internet-Draft, draft-ietf-anima-constrained-voucher-16, 14 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-anima-constrained-voucher-16.txt>>.
- [I-D.kuehlewind-update-tag]  
Kuehlewind, M. and S. Krishnan, "Definition of new tags for relations between RFCs", Work in Progress, Internet-Draft, draft-kuehlewind-update-tag-04, 12 July 2021, <<https://www.ietf.org/archive/id/draft-kuehlewind-update-tag-04.txt>>.
- [I-D.richardson-anima-voucher-delegation]  
Richardson, M. and W. Pan, "Delegated Authority for Bootstrap Voucher Artifacts", Work in Progress, Internet-Draft, draft-richardson-anima-voucher-delegation-03, 22 March 2021, <<https://www.ietf.org/archive/id/draft-richardson-anima-voucher-delegation-03.txt>>.
- [onpath] "can an on-path attacker drop traffic?", n.d., <<https://mailarchive.ietf.org/arch/msg/saag/mlr9uo4xYznOcf85EyK0Rhut598/>>.
- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, DOI 10.17487/RFC5652, September 2009, <<https://www.rfc-editor.org/info/rfc5652>>.
- [RFC8792] Watsen, K., Auerswald, E., Farrel, A., and Q. Wu, "Handling Long Lines in Content of Internet-Drafts and RFCs", RFC 8792, DOI 10.17487/RFC8792, June 2020, <<https://www.rfc-editor.org/info/rfc8792>>.

- [RFC8812] Jones, M., "CBOR Object Signing and Encryption (COSE) and JSON Object Signing and Encryption (JOSE) Registrations for Web Authentication (WebAuthn) Algorithms", RFC 8812, DOI 10.17487/RFC8812, August 2020, <<https://www.rfc-editor.org/info/rfc8812>>.
- [RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/info/rfc8949>>.

## Appendix A. Examples

These examples are folded according to [RFC8792] Single Backslash rule.

### A.1. Example Pledge Voucher Request - PVR (from Pledge to Registrar)

The following is an example request sent from a Pledge to the Registrar, in "General JWS JSON Serialization".

```
{
  "payload":
    "eyJpZXRmLXZvdWNoZXItcmVxdWVzdDp2b3VjaGVyIjp7ImNyZWZlZWQtb24iOiIyMDE5LTAyLTE4VDA3OjM5OjAzLjAwMzoiLCJub25jZSI6IjU3NDI2OTg0MjI2ODAwNzIifX0",
  "signatures": [
    {
      "protected":
        "eyJhbGciOiJFUzI1NiIsIng1YyI6WyJNSU1CMmpDQ0FZQ2dBd0lCQWdJR0FXZWdkY1NMTUFvR0NDcUdTQTQ5QkFNQ01EMHhDekFKQmdOVkJBWVRBa0ZSTVJVd0V3WURWUVFLREF4S2FXNW5TbWxlWjB0dmNuQXhGekFWQmdOVkJBWVRBa0ZSTVJVd0V3WURWUVFLREF4S2FXNW5WRlZ6ZEV0Qk1DQVhEVEU0TVRJeE1qQXpNamcxTVZvWUR6azVPVGt4TWpNeE1qTTFPFVU1V2pCU01Rc3dDUVlEVlFRR0V3SkJVVVEVWTUJNR0ExVUVDZ3dNU21sdVowcHBibWREYjNkD01STXdFUVlEVlFRRkV3b3dNVE16TkrVMk56ZzVNUmN3RlFZRFZRUUREQTVLYVc1b1NtbHVhMFJsZG1salpUQ1pNQk1HQnlxR1NNND1BZ0VHQ0Nxr1NNND1Bd0VIQTBjQUJNVkdHOE1cGpmNWpYbnlyVXJYeVoxa1BncUJlM05YdTFkVEFEZStyL3Y2SnpsJGwzNTVJZ2NIQzNheHBpYnFKTS9iV1JhRXlqcWNDSmo0akprb3dDdWpWVEJUTUN3R0NTc0dBVFCZ3U1U0FnUWZlEQJf0WVhOaExYUmxjM1F1YzJsbGJXVnVjeTFpZEM1dVpYUTZPVFEwTXpBVEJnTlZlU1VFRERBS0JnZ3JCZ0VGQ1FjREFqQU9CZ05WSFE4QkFmOEVCQU1DQjRlBd0NnWU1Lb1pJemowRUF3SURTQUF3U1FjZ1d0UHpJSVhZMm14U1hKdEV4S0VoaFpkYTRYK0VwbFpvbUVJMNpBMGRzam9DSVFDM0pwUW1SWE1Hbi9wNEJlOW16aWk5MmVjbFR4NC9PNHJsbTdNeUxxa2hkQT09I119",
      "signature":
        "xURZmcWSFaBD2cNkr37azT9osWfzTZ_veCsVho3fwdD6NR4ghL61VJmY_ra0a42SvoW2Tu4X1ldzzD8VDtCCDg"
    }
  ]
}
```

Figure 2: Example Pledge Voucher Request - PVR

## A.2. Example Parboiled Registrar Voucher Request - RVR (from Registrar to MASA)

The term parboiled refers to food which is partially cooked. In [BRSKI], the term refers to a Pledge voucher-request (PVR) which has been received by the Registrar, and then has been processed by the Registrar ("cooked"), and is now being forwarded to the MASA.

The following is an example Registrar voucher-request (RVR) sent from the Registrar to the MASA, in "General JWS JSON Serialization". Note that the previous PVR can be seen in the payload as "prior-signed-voucher-request".

```

{
  "payload":
    "eyJpZXRmLXZvdWNoZXItcmVxdWVzdDp2b3VjaGVyIjpp7InNlcm1hbC1udWliZXIiOiIwMTIzNDU2Nzg5Iiwibm9uY2UiOiI1NzQyNjk4NDIyNjgwNDcyIiwicHJpb3Itc2lnbmVkbXZvdWNoZXItcmVxdWVzdCI6ImV5SndZWGxzYjJGa0lqb2laWGxLY0ZwWVtMU1XRnAyWkZkt2IxcFlTWfJqYlZaNFpGZFd1bVJFY0RkaU0xWnFZVWRXZVZVscWNEZEpiVTU1V2xkR01GcFhVWFJpTWpScFQybEp1VTFFU1RWTvZFRjVUR1JGtKZaRVFUT1BhazAxVDJwQmVreHFRWGRUUm05cFRFTktkV015TldwYVUwazJTV3BWTtA1RVNUS1BWR2N3VFdwSk1rOUVRVEJPZWtscFpsZ3dJaXdpYzJsbmJtRjBkWEpsY3lJN1czc2ljSEp2ZEdWamRHVmtJam9pWlhsS2FHSkhZMmxQYVWwR1ZyCEpNVTvU1hOSmJtY3hXWGxKTmxkNVNrNVRWV3hEVFcxd1JGRXdsbHBSTW1SQ1pEQnNRMUZYWkVwU01FW1lXbGRrYTFreFRrMVVWVWoyVWpCT1JHTlZaR1JVVkZFMVvXdeUebEV3TVVWTLNHaEvaV3RHUzFGdFpFOvdhMHBdVjFaU1FtRXdxXbE5VVmtwV1pEQldNMWRWVWxkV1ZrWk1Va1ZHTkZNeVJsaE9WelZVWwXkNE1WZHFrazlrYlU1MVVWaG9SM1ZyUmxkUmJXU1BwbXRLUWxSVk1VmhNMEozV1cweGEXtX1SbGhPVnpWWFVqRmFobHBGVms5UmF6RkVVVlpvU1ZaR1ZUQ1VWbEpLW1VVeGNWR1ljRTVoYldONFZGWMfkbGRWVWpaaGVSWlFWa2QwTkZSWGnFNWxSVEZ4VkJZSR1VGWkdWVEZXTW5CRFZUQXhVbU16WkVSV1ZteEZWbXhHVWxJd1ZqTlRhMHBXVmtWV1YxU1ZTazVTUUVWNFZsV1dSRm96WkU1Vk1qRnpaRlp2ZDJOSVftbG1WMUpGV1dwT1MyUXdNVk5VV0dSR1ZWWnNSVlpzUmxKU2ExWXpZak5rVGxaRmJEW1VhMUpXVfdzMU5scDZWazVWY1U0elVteEdXbEPHV2xKV1ZWSkZVV1JXVEZsV1l6RmlIRTUwWWtoV1lVMUdTbk5hUnpGellXeHdWVkJZzY0U1UmF6Rk1VVzVzZUZJeFRrNU9SR3hDV2pCV1NGRXdUbmhTTVU1T1RrUnNRbVF3VmtsU1ZfSktVV1ZLVGxac1pFaFBsbTh4WTBkd2JVNvhjRmxpYm14NVZsaEtXV1ZXyJNoaE1VSnsVZMVZLYkUwd05WbGtWRVpyVmtWR1JWcFRkSGxNTTfreVUyNXdTbE5IZDNwT1ZGWktXakpPU1ZGN1RtaGxTRUp3V1c1R1MxU1RPV2xXTVVwb1VsaHNjV05YVGtSVGJXOHdZV3R3Y21JelpFUmtWM0JYVmtWS1ZWU1ZUak5TTU1V1V16QmtRbFZXUmtOU0xVXhWVEJHYmxWWFdrV1Jha1l3VjFab1QyRkZ1RmxWY1hocVRURkdNVmw2U25OaVIwcFlWbTVXYW1WVWJuQmFSVTB4WkZad1dWV1VXbEJXUmtWM1ZGaHdRbFpGU201VWJGcEpWVEZXUmxKR1VrSlRNRXB1V2pOS1Exb3dWa2RSYkVacVVRVkdjVkJZWT1VOYU1EV1hVMFpGtKZGclJtMVBSVlpEVVZVeFJGRnFVa0prTUU1dVYxVnNUR014Y0VwbGJXOTNVbFZHTTFOV1VsU1JWV16Vld4R1Nsb3haREJWU0hCS1UxWm9XazF0YkRSVmjHaExaRVZXTkZNd1ZtOWhSbkJyV1ZSU1dVc3dWbmRpUm5CM1lsVldTazF1Y0VKt1IxSjZZVzA1UkZOV1JrUk5NSEIzV1ZjeFUxZEZNVWhpYVRsM1RrVktNVT1YYkRaaFYyczFUVzFXYW1KR1VqUk9RemxRVGtoS2MySlVaRTVsV1hoNF1USm9hMUZVTURsSmJERTVJaXdpYzJsbmJtRjBkWEpsSWpvaWVGv1NXbTFqVjFOR1lVSkVNbU5PYTNJek4yRjZWrgx2YzFkbWVsUmFYm1psUTNOV2FHOHpabmRrUkRaT1VqUm5hRXcyTVZaS2JWbGZjbUV3WVRReVUzWnZWekpVZFRSwwJHeGt1bnBFT0ZaRWRFTkRSR2NpZlYxOSIsImNyZWf0ZWQt24iOiIyMDIyLTazLTayVDAzOjAxOjI0LjQ2N1oifX0",
  "signatures": [
    {
      "protected":
        "eyJ4NWMiOl5iU1JQm96Q0NBVXFhQXZdJQkFhNSUdBVzBlTHVJRk1Bb0d"
    }
  ]
}

```

```

DQ3FHU000OUJBTUNNRfV4RXpBUkJnTlZCQW9NQ2sxNVFuVnphVzVsYzN
NeERUQUxCZ05WQkFjTUJGTnBkR1V4RHpBTkJnTlZCQU1NQmxSbGMzUkR
RVEFlRncweE9UQTVNVEV3TWpNM016SmFGdzB5T1RBNU1URXdNak0zTXp
KYU1GUXhFekFSQmdOVkJBb01DazE1UW5WemFXNWxjM014RFRBTEJnTlZ
CQWNNQkZOCGRHVXhMakFzQmdOVkJBTU1KVkpsWjJsemRISmhjaUJXYjN
WamFHVnlJRkpsY1hWbGMzUWdVMmxuYm1sdVp5QkxaWGt3V1RBVEJnY3F
oa2pPUFFJQkJnZ3Foa2pPUFFNQkJ3TkNBQVQ2eFZ2QXZxVHoxW1VpdU5
XaFhwUXNrYVB5N0FISFFMdlhpSjBpRUx0NnVOUGFuQU4wUW5XTV1PXC8
wQ0RFaklrQlFvYnc4WUtXanR4SkhWU0dUajlLT295Y3dKVEFUQmdOVkh
TVUVEREFLQmdnckJnRUZCUWNESEERBT0JnTlZiUTHCQWY4RUJBTUNCNEF
3Q2dZSUtvWk16ajBFQXdJRFJ3QXdSQUlnWXIyTGZxb2FDS0RGNFJBY01
tSmkrTkNacWRtaXVWdWdJU0E3T2hLUneEzWUNJRHHuUE1NbnBYQU1Uc1B
KdVBXewNlRVIXMVB4SE9uKzBdcFNiATJxZ3BXWCIsIk1JSUJwRENDQV
tZ0F3SUIJBZ01HQVcwZUx1SCtNQW9HQ0Nxr1NNND1CQU1DTURVeEV6QVJ
CZ05WQkFvTUNrMTVRblZ6YVc1bGMzTXhEVEFMQmdOVkJBY01CRk5wZEd
VeER6QU5CZ05WQkFNTUJJsUmxiM1JEUVRBZUz3MHhPVEE1TVRFd01qTTN
NekphRncweU9UQTVNVEV3TWpNM016SmFNRFV4RXpBUkJnTlZCQW9NQ2s
xNVFuVnphVzVsYzNNeERUQUxCZ05WQkFjTUJGTnBkR1V4RHpBTkJnTlZ
CQU1NQmxSbGMzUkRRVEJaTUJNR0J5cUdTTTQ5QWdFR0NDcUdTTTQ5QXd
FSEEWsUUFCT2t2a1RldThRbFQzRkhKMVhSTcrV3NIT2IwVVMzU0FMdEc
1d3VLUURqaWV4MDZcLlNjWTVQSmlidmdIVEIrrLwvUVRqZ2VsSEd5MV1
LcHdjTk1jc1N5YWpSVEJETUJJR0ExVWRfV0V3CXC93UU1NQV1CQWY4Q0F
RRXdeZ1lEV1IwUEFRSfFwvQkFRREFnSUVNQjBHQTfVZERnUVdCQ1RvWk1
NelFkc0RcL2pcLytnWFwvN2NCSnVjSFwvWG1qQUtCZ2dxaGtqT1BRUUR
BZ05KQURCR0FpRUf0eFEzK0lMR0JQSXRtaDRiOVdYaFhOdWhxU1A2Sct
iXC9MQ1wvZ1ZzRGpRNm9DSVFERzJ1UkNiBFZmxM3loQjU4VFhNVWJ6SDg
rT2xoV1V2T2xSRDNWRXFEZGNRdz09Il0sImFsZyI6IkVTMjU2In0",
"signature":
  "zvtnaEDpOqL49XnYVRbLxVAaZCMRtDiaLqMeFSh3UsjHdz4FT0lFywV
  7-5inMpafXTnqxnD2Gpr3ClUXUyAJg"
  }
}

```

Figure 3: Example Parboiled Registrar Voucher Request - RVR

## A.3. Example Voucher Response (from MASA to Pledge, via Registrar)

The following is an example voucher response from MASA to Pledge via Registrar, in "General JWS JSON Serialization".

```

{
  "payload":
    "eyJpZXRMXXZvdWNoZXI6dm91Y2hlciI6eyJhc3NlcnRpb24iOiJsb2
    dnZWQ1LCJzZXJpYWwtbnVtYmVyIjoieMDEyMzQ1Njc4OSIsIm5vbmNlI
    joiNTc0MjY5ODQyMjY4MDQ3MiIsImNyZWZlZWQtb24iOiIyMDIyLTAz
    LTAyVDAzOjAxOjI0LjYxOFoiLCJwaW5uZWQtZG9tYWluLWNlcnQiOiJ
    NSU1CcERDQ0FVbWdDd01CQWdJR0FXMGVmdUgrTUFvR0NDcUdTtTQ5Qk
    FNQ01EVXhFekFSQmdOVkJBb01DazE1UW5WemFXNWxjM014RFRBTEJnT
    lZCQWNNQkZ0cGRHVXhEekFOQmdOVkJBtU1CbFJsYzNSRFFUQWVGdzB4
    T1RBNU1URXdNak0zTXpKYUZ3MHlPVEE1TVRFd01qTTNnekphTURVeEV
    6QVJCZ05WQkFvTUNrMTVRblZ6YVc1bGMzTXhEVEFMQmdOVkJBjY01CRk
    5wZEdVeER6QU5CZ05WQkFNTUJzUmxiMjE1JEUVRWk1CTUdCeXHU0000
    UFnRUdDQ3FHU000OUF3RUhBME1BQk9rdmtUSHU4UWxUM0ZISjFVYUk3
    K1dzSE9iMFVTM1NBTHRHNXd1S1FEamlleDA2L1NjWTVQSmliZmVlVEI
    rRi9RVGpnZWxIR3kxWUtd2NOTWNzU3lhalJlUQkRNQk1HQTFVZEV3RU
    Ivd1FJTUFZQkFmOENBUUV3RGdZRFZSMFBBUgVqFRREFnSUVNQjBHQ
    TFVZERnUVdCQ1RvWk1NelFkc0Qvaih8rZl9vN2NCSnVjSC9YbWpBS0Jn
    Z3Foa2pPUFFRREFnTkpBREJHQLFQXR4UTMrSUxHQ1BJdFNoNGI5Vlh
    oWE5laHFTUDZIK2IvTEMvZlZlZGRGpRNm9DSVFERzJlUkNlbnFzXmM3loQj
    U4VFhNVWJ6SDgrT2xoV1V2T2xSRDNWRXFEZGNRdz09In19",
  "signatures": [
    {
      "protected":
        "eyJ4NWMiOiJsiTUlJQmt6Q0NBVGlnQXhJQkFnSudBV0ZCakNrWU1B
        b0dDQ3FHU000OUJBTUNNRDB4Q3pBSklnTlZCQVlUQWtGUk1SVXdfd
        11EVlFRS0RBeEthVzVuU21sdVowInZjbkF4RnpBVklnTlZCQU1NRG
        twcGJtZEthVzVuVkdWemRFTklnQjRFRFRFNE1ERXlPVEV3TlRjME1
        GblhEVEk0TURFeU9URXdOVEkwTUZvd1R6RUxNQWtHQTFVRUJJoTUNR
        VkV4RlRBVEJnTlZCQW9NREVwcGJtZEthVzVuUTI5eWNERXBNQ2NHQ
        TFVRUF3d2dTbWx1WjBwcGJtZERiM0p3SUZadmRXIm9aWElnVTJsbm
        JtbHVaeUJlMlhrdldUQVRCZ2NxaGtqTlBRUjZCZ2dxaGtqTlBRTUJ
        Cd05DQUFTQzZiZUxBbWVxMVZ3Nm1RclJzOFIwW1crNGIxR1d5ZG1X
        czJHQU1GV3diaXRmMm5JWEgzT3FIS1ZlOHMyUnZpQkd0aXZPS0dCS
        Eh0QmRkRkVaWnZiN294SXdfFREFPQmdOVkhROEJBZjhFQkFNQ0I0QX
        dDZl1JS29aSXpqbEVbd01EU1FBd1JnSWhBSTRQWWJ4dHNzSFAyVkh
        4XC90e1VvUVVvU3N5ZEwzMERRSU5FdGNOOW1DVfHQW1FQXZJYjNv
        K0ZPM0JUBmNMRnNhSlpSQWtkN3pPdXNuXC9cL1pLT2FFS2JzVkrpV
        T0iXSwiYWxnIjoieRVMYNTYifQ",
      "signature":
        "vyge3GENm1BNcijXT5VH7A8CJWW7wPzH61u2VCfR8E9v8H8Yr3g9
        irYz4q5sYj2UnOVih-hG_ogrZR0Tct_Vzw"
    }
  ]
}

```

Figure 4: Example Voucher Response

Authors' Addresses

Michael Richardson  
Sandelman Software Works  
Email: mcr+ietf@sandelman.ca

Thomas Werner  
Siemens AG  
Email: thomas-werner@siemens.com

anima  
Internet-Draft  
Intended status: Standards Track  
Expires: December 30, 2021

Y. Li  
L. Shen  
Huawei Technologies  
June 28, 2021

Autonomic IP Address To Access Control Groups Mapping  
draft-yizhou-anima-ip-to-access-control-groups-00

Abstract

This document defines the autonomic technical objectives for IP address/prefix to access control groups mapping. The objectives defined can be used in Generic Autonomic Signaling Protocol (GRASP) to make the policy enforcement point receive IP address and its tied access control groups information directly from the access authentication points and then execute the group based policies.

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 30, 2021.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminologies . . . . .	3
3. Problems . . . . .	3
4. Autonomic IP Address to Access Control Groups Mapping Procedures . . . . .	6
4.1. Behaviours of IP Address to Access Control Groups Mapping Requesting Nodes . . . . .	6
4.2. Behaviours of IP Address to Access Control Groups Mapping Providing Nodes . . . . .	7
5. Autonomic IP Address to Access Control Groups Objectives . .	8
5.1. IPAddressToAccessControlGroups Objective Option . . . . .	8
6. Security Considerations . . . . .	9
7. IANA Considerations . . . . .	10
8. Acknowledgements . . . . .	10
9. References . . . . .	10
9.1. Normative References . . . . .	10
9.2. Informative References . . . . .	10
Appendix A. Objective Examples . . . . .	12
Authors' Addresses . . . . .	13

## 1. Introduction

Ubiquitous group based policy management makes sure that the users can obtain the same network access permission and QoS assurance wherever they access the campus network. That is, the permission and QoS assurance are tied to user role, rather than access points and/or IP address assigned.

Group means a number of endpoints connecting to the network that share common network policies. It facilitates the easy design and provision of policy. A user's role is usually a group indicated by a group ID. Group based policy management has been replacing the traditional IP address and/or port number based policy widely.

The policy enforcement point (PEP) requires the IP address/prefix and access control group mapping information of user in order to execute the group based policy. This mapping information is usually first available at the access authentication point (AAP) during the procedures of user access and authentication/authorization. However PEP may not be the access authentication point. Therefore IP and group mappings has to be passed to PEP.

This document defines the autonomic technical objectives for IP address/prefix and access control group mapping. In this document, group is also used for short to refer to the access control group. The Generic Autonomic Signaling Protocol (GRASP) [RFC8990] can make use of these technical objectives as the basic building blocks of a ubiquitous group based policy management solution, especially for a campus network.

Autonomic Networking Infrastructure (ANI) is designed to provide the elementary functions and services to be further integrated and used by Autonomic Service Agents (ASA) on nodes. The campus policy management ASA can integrate the function introduced in this document when necessary.

## 2. Terminologies

This document uses terminology defined in [RFC7575].

PEP: Policy Enforcement Point. A logical entity that enforces policy decisions [RFC3198]. The policy decisions are group based policies in this document.

AAP: Access Authentication Point. A logical entity that obtains the information of the attaching clients' assigned IP address/prefix and their access control groups. AAP may get the information from one or different resources, for example, DHCP [RFC2131] [RFC8415] server and/or RADIUS [RFC3198] server.

## 3. Problems

The traditional policy in a campus network is normally presented as IP prefix/address based, for example, "Deny the traffic from IP prefix X to IP prefix Y". Each of the access port of the switches is assigned a subnet prefix and each subnet implies a group. It works well when the end hosts are static. With the increasing deployment of wireless accessed users and more complicated and dynamic requirements of campus network policy, such an assumption no longer hold. For instance, a user from the engineering department may bring the laptop to access the campus network via a WiFi access point. Then it will be assigned an IP address from a different subnet prefix from the other fixed end hosts in the same engineering department. It is hard and tedious to provision the consistent policy with the other hosts in the same group for this specific IP address. Another example is a user can belong to more than one group, say group of department A and also VIP group. Group assignment is much more flexible than subnet defined IP address assignment.

Therefore group based policy is used in such cases. No matter what IP address is assigned to the user, its belonging access control groups have no change and the group based policies has no change too. For example, the policy can be "Allow the traffic from group engineering to group testing, and assign the traffic destined to VIP group the highest priority". In order to make group based policy work, the IP address and its group mapping information has to be stored on PEP so that IP addresses carried in data packet can be mapped to the group ID first and then the policy can be enforced.

IP and group mapping information is usually first available at the access authentication point (AAP). Figure 1 show a typical campus network. The policy enforcement point (PEP) can be core switches, while the access authentication point (AAP) is the access switch in the figure. AAP serves as the DHCP relay which remembers the IP address assigned to the client and/or at the same time it talks to AAA server to get the client's group information based on client's identity.

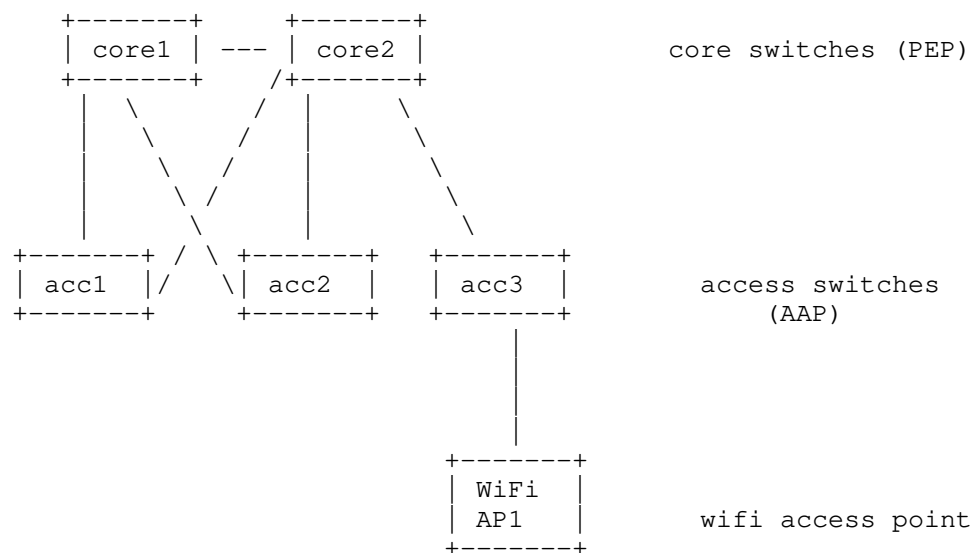


Figure 1: Hierarchical Campus Network

A more complex campus network is shown in Figure 2. There are 4 PEPs are deployed at the key positions for different types of traffic. The AAPs obtaining a user's IP and group ID mapping information are access switches (not fully shown in the figure) which are the access nodes for the attaching clients.

The problem to be solved by Autonomic Networking here is how to make IP address/prefix and access control group mapping information available at PEP from AAP.

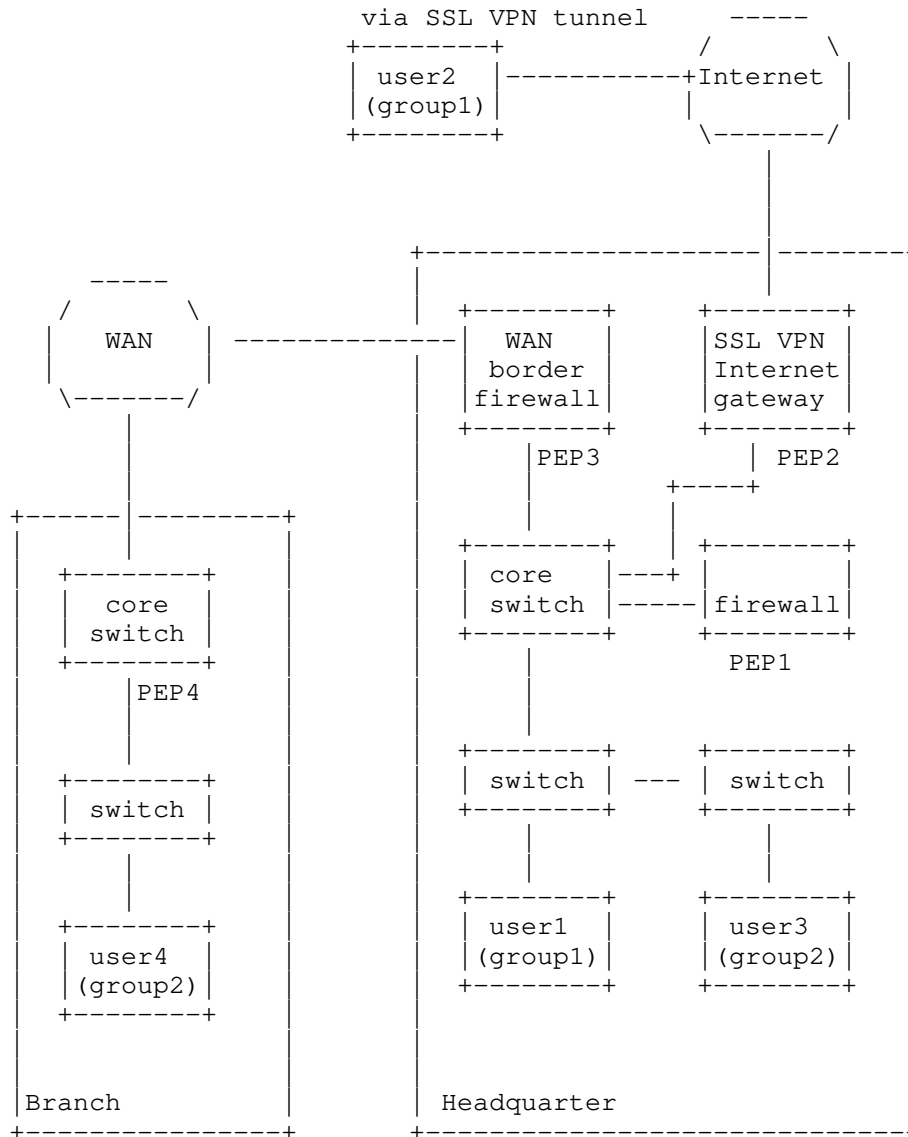


Figure 2: Campus Networks with remote access

Some deployment uses a centralized controller to solve this problem. Every single AAP reports its IP and group ID mapping information to the controller. When a PEP receives a data packet, it queries the controller for the group IDs of the source and/or destination IP addresses and then enforce the group based policy. This approach requires an explicit controller able to talk to each and every AAP and PEP. In the deployment where the headquarter and branch campus networks are far apart, it will require controllers for each site to exchange information or have another super-controller to help exchange the information among sites. It introduces the complexity and interoperability issues.

Autonomic Networking (AN) puts the intelligence at the node level, to minimize dependency on human administrators and central management such as a controller. The Autonomic Networking approach discussed in this document is based on the assumption that there is a generic discovery and negotiation protocol that enables direct negotiation between the routers or switches. GRASP [RFC8990] is intended to be such a protocol which can make use of the technical objectives defined in the following sections as the basic building blocks of a ubiquitous group based policy management solution, especially for a campus network. The ultimate goal is self-management of campus networks which can expand over multiple sites and share the same set of policies, including self-configuration, self-optimization, self-healing and self-protection (sometimes collectively called self-X).

#### 4. Autonomic IP Address to Access Control Groups Mapping Procedures

An Autonomic Service Agent (ASA) participates in IP address/prefix to access control groups mapping is called IPAddressToAccessControlGroups ASA in this document. The procedures carried out by IPAddressToAccessControlGroups ASA is illustrated below.

##### 4.1. Behaviours of IP Address to Access Control Groups Mapping Requesting Nodes

IPAddressToAccessControlGroups requesting node is usually a PEP in a domain which executes the group based policy. So it needs to map an IP address/prefix to one or more group IDs first. Such mapping information will be stored locally until either timeout or withdrawn.

The request can be triggered by a data packet. Group based policy requires both the source and destination group IDs which are normally mapped from source and destination IP addresses. If any of such mapping is not locally available, the requesting node needs to ask for it. In some implementation, data packet encapsulation includes the source group ID directly such as in the reserved field in VXLAN

[RFC7348]. Therefore it is up to the requesting node to determine if both source and destination groups or only one of them to be queried. In some cases that the requesting node is a tunnel endpoint, it should be noted that usually the inner rather than outer IP addresses are to be used to query for the corresponding group id.

The request can also be sent periodically or voluntarily. It can happen when a newly booted requesting node wants to get the whole batch of IP address and access control group mapping information or when a requesting node would like have an explicit refreshment on the information.

The IPAddressToAccessControlGroups ASA should send out a GRASP Discovery message that contains a IPAddressToAccessControlGroups Objective option in order to discover peers supporting this option. The requesting ASA acts as a GRASP synchronization initiator by sending a GRASP Request message with a IPAddressToAccessControlGroups Objective option. The ASA indicates an IP prefix or address in this option. This starts a GRASP synchronization process.

#### 4.2. Behaviours of IP Address to Access Control Groups Mapping Providing Nodes

IPAddressToAccessControlGroups providing node is usually an AAP of a user in a domain. It obtains the mapping of IP address and group IDs of an endpoint in various ways. For instance, use RADIUS [RFC2865] or CAPWAP [RFC5415] to get the user's access control group IDs during authentication phase and/or use DHCP snooping to get the user's assigned IP address. Sometimes such mapping information can be statically provisioned based on port or VLAN. The mapping information is stored locally on AAP.

A device that receives a Discovery message with a IPAddressToAccessControlGroups Objective option should respond with a GRASP Response message if it contains a IPAddressToAccessControlGroups ASA. When this ASA receives a subsequent Request message, it should reply with a GRASP Synchronization messages. The Synchronization messages carry a IPAddressToAccessControlGroups Objective option, which will indicate the mappings between the IP address/prefix and group IDs. Optionally the expiration time can be tied to each mapping.

The IP address to access control groups mapping providing node can send the Flood Synchronization message if it has any update or withdraw regarding its providing mapping information.

The providing nodes of address to access control groups mapping information are usually at the edges and can be added or replaced

with the expansion of the network. The requesting nodes are normally aggregation or core nodes with more storage and capability to enforce the policy. Therefore the number of mapping information providing nodes is usually more than the number of requesting nodes. The providing node can be the one initiating the GRASP Discovery message as well and/or send the unsolicited Synchronization message [I-D.ietf-anima-grasp-distribution].

## 5. Autonomic IP Address to Access Control Groups Objectives

This section defines the GRASP technical objective options that are used to support autonomic IP address/prefix to access control groups mapping.

### 5.1. IPAddressToAccessControlGroups Objective Option

The IPAddressToAccessControlGroups Objective option is a GRASP objective option conforming to [RFC8990]. The name of this option is "IPAddressToAccessControlGroups". It carries the IP prefix/address and its mapping access control group id. The format of IPAddressToAccessControlGroups Objective option in CBOR (Concise Binary Object Representation [RFC8949]) is show in Concise data definition language (CDDL) [RFC8610] as follows. Tags for general IPv4 and IPv6 addresses and prefixes defined in [I-D.ietf-cbor-network-addresses] are used.

```
objective = ["IPAddressToAccessControlGroups",
             objective-flags, loop-count,
             [ip-address-or-prefix, *group-id]]

group-id = uint

; copied from draft-ietf-cbor-network-addresses, RFC YYYY TBD:

ip-address-or-prefix = ipv6-address-or-prefix/ipv4-address-or-prefix

ipv6-address-or-prefix = #6.54(ipv6-address / ipv6-prefix)
ipv4-address-or-prefix = #6.52(ipv4-address / ipv4-prefix)

ipv6-prefix = [ipv6-prefix-length, ipv6-prefix-bytes]
ipv4-prefix = [ipv4-prefix-length, ipv4-prefix-bytes]

ipv6-prefix-length = 0..128
ipv4-prefix-length = 0..32

ipv6-prefix-bytes = bytes .size (uint .le 16)
ipv4-prefix-bytes = bytes .size (uint .le 4)

ipv6-address = bytes .size 16
ipv4-address = bytes .size 4

; copied from the GRASP specification, RFC 8990:

objective-flags = uint .bits objective-flag

objective-flag = &(amp;
  F_DISC: 0      ; valid for discovery
  F_NEG: 1       ; valid for negotiation
  F_SYNCH: 2     ; valid for synchronization
  F_NEG_DRY: 3   ; negotiation is a dry run
)
loop-count = 0..255
```

A common practice currently usually uses 16 bits to present a group ID. But the representation does not limit that. Zero group ID would be used for full retraction of a prefix or address.

## 6. Security Considerations

Security consideration for GRASP [RFC8990] applies in this document. The preferred security model is that devices are trusted following the secure bootstrap procedure [RFC8995] and that a secure Autonomic Control Plane (ACP) [RFC8994] is in place.

## 7. IANA Considerations

This document defines a new GRASP Objective option name: "IPAddressToAccessControlGroups". The IANA is requested to add it to the "GRASP Objective Names" subregistry defined by [RFC8990].

## 8. Acknowledgements

Thanks to Carsten Bormann, Brian Carpenter and Michael Richardson for useful suggestions and revising CDDL representations.

## 9. References

### 9.1. Normative References

- [RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <<https://www.rfc-editor.org/info/rfc7575>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.
- [RFC8990] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <<https://www.rfc-editor.org/info/rfc8990>>.
- [I-D.ietf-anima-grasp-distribution] Liu, B., Xiao, X., Hecker, A., Jiang, S., Despotovic, Z., and Brian, "Information Distribution over GRASP", draft-ietf-anima-grasp-distribution-02 (work in progress), March 2021.
- [I-D.ietf-cbor-network-addresses] Richardson, M., "CBOR tags for IPv4 and IPv6 addresses and prefixes", draft-ietf-cbor-network-addresses-04 (work in progress), April 2021.

### 9.2. Informative References

- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<https://www.rfc-editor.org/info/rfc2131>>.

- [RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <<https://www.rfc-editor.org/info/rfc2865>>.
- [RFC3198] Westerinen, A., Schnizlein, J., Strassner, J., Scherling, M., Quinn, B., Herzog, S., Huynh, A., Carlson, M., Perry, J., and S. Waldbusser, "Terminology for Policy-Based Management", RFC 3198, DOI 10.17487/RFC3198, November 2001, <<https://www.rfc-editor.org/info/rfc3198>>.
- [RFC5415] Calhoun, P., Ed., Montemurro, M., Ed., and D. Stanley, Ed., "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification", RFC 5415, DOI 10.17487/RFC5415, March 2009, <<https://www.rfc-editor.org/info/rfc5415>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.
- [RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <<https://www.rfc-editor.org/info/rfc8949>>.
- [RFC8994] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "An Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <<https://www.rfc-editor.org/info/rfc8994>>.
- [RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <<https://www.rfc-editor.org/info/rfc8995>>.

## Appendix A. Objective Examples

This appendix shows a number of examples of objective defined in this document conforming to the CDDL syntax given in Section 5.1.

```
["IPAddressToAccessControlGroups", 15, 101,  
  [54([4, h'A50386A78BA56FA4BBC734281C51']), 3506, 2698, 4562]]  
  
["IPAddressToAccessControlGroups", 5, 73, [52(h'9946B8A3'), 2881,  
  2265, 1720, 2450]]  
  
["IPAddressToAccessControlGroups", 15, 161,  
  [54(h'39F3045B641AD291B057CD1857A7314A')]]  
  
["IPAddressToAccessControlGroups", 15, 2, [52(h'98A1CE4F')]]  
  
["IPAddressToAccessControlGroups", 15, 66, [52(h'69A16BFE'), 2601,  
  1851, 3876, 1405]]  
  
["IPAddressToAccessControlGroups", 15, 254,  
  [54(h'38AB303B8895DC95068CE00248D2FE91'), 4019, 1166, 3113]]  
  
["IPAddressToAccessControlGroups", 15, 63, [52([4, h'0B48']), 3035,  
  1181]]  
  
["IPAddressToAccessControlGroups", 15, 44, [52(h'01F1D8FF'), 3099,  
  1577, 1138, 1670]]  
  
["IPAddressToAccessControlGroups", 15, 181,  
  [54(h'2C74719F9355BA4E3BDE5689D1FE4CB0')]]  
  
["IPAddressToAccessControlGroups", 15, 129, [52(h'A2EF97C7'), 3149,  
  2728]]  
  
["IPAddressToAccessControlGroups", 15, 18,  
  [54(h'CD3868615B00D72A61A028822FEE6407'), 1832, 4605, 360, 3030]]  
  
["IPAddressToAccessControlGroups", 15, 171,  
  [54(h'46929AE1103FDF6407A239323F71C234')]]  
  
["IPAddressToAccessControlGroups", 15, 42, [52([7, h'8E05')]]]  
  
["IPAddressToAccessControlGroups", 15, 180,  
  [54([41, h'2D85855FA9C3772AAB2F']), 672, 1205]]
```

Authors' Addresses

Yizhou Li  
Huawei Technologies

Email: liyizhou@huawei.com

Li Shen  
Huawei Technologies

Email: kevin.shenli@huawei.com