DNS-SD Compatible Service Discovery in GeneRic Autonomic Signaling Protocol (GRASP)
draft-eckert-anima-grasp-dnssd-03

Abstract

DNS Service Discovery (DNS-SD) defines a framework for applications to announce and discover services. This includes service names, service instance names, common parameters for selecting a service instance (weight or priority) as well as other service-specific parameters. For the specific case of autonomic networks, GeneRic Autonomic Signaling Protocol (GRASP) intends to be used for service discovery in addition to the setup of basic connectivity. Reinventing advanced service discovery for GRASP with a similar set of features as DNS-SD would result in duplicated work. To avoid that, this document defines how to use GRASP to announce and discover services relying upon DNS-SD features while maintaining the intended simplicity of GRASP. To that aim, the document defines name discovery and schemes for reusable elements in GRASP objectives.

Note to the RFC Editor

Please replace all occurrences of rfcXXXX with the RFC number assigned to this document.

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
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. Overview .................................................. 3
2. Terminology ................................................. 4
3. Specification ................................................ 4
   3.1. Service and Name Objectives ........................... 4
   3.2. Objective Value Reuseable Elements Structure .......... 5
   3.3. Reuseable Elements ................................... 6
      3.3.1. Sender Loop Count ............................... 6
      3.3.2. Service Element ................................ 6
      3.3.3. Name Element ................................... 9
4. Theory of Operation ........................................ 11
   4.1. Using GRASP Service Announcements .................... 11
   4.2. Further Comparison with DNS-SD ....................... 13
   4.3. Open Issues ........................................ 13
5. Security Considerations .................................... 14
6. IANA Considerations ....................................... 14
7. Acknowledgements ......................................... 15
8. Contributors .............................................. 15
9. Change log [RFC Editor: Please remove] ................... 15
   9.1. 03 - Refresh ......................................... 15
   9.2. 02 - Revived after charter round 1 finished .......... 15
   9.3. 01 - ................................................ 15
   9.4. 00 - Initial version ................................ 15
10. References ................................................ 15
   10.1. Normative References ................................. 15
   10.2. Informative References .............................. 16
Authors’ Addresses .......................................... 16
1. Overview

DNS Service Discovery (DNS-SD) [RFC6763] defines a framework for applications to announce and discover services. This includes service names, service instance names, common parameters for selecting a service instance (weight, priority) as well as other service-specific parameters.

GeneRic Autonomic Signaling Protocol (GRASP) [RFC8990] is intended to also be used for service discovery purposes. Reinventing service discovery for GRASP with a similar set of features would result in duplication of work. Therefore, this document defines how to use GRASP to announce and discover services in a way that inherits DNS-SD features and also tries to be compatible in spirit as much as possible while still maintaining the intended simplicity of GRASP.

The goal of this document is to permit defining service and their parameters once and then use that in GRASP, mDNS and (unicast) DNS. Future work can also define DNS-SD <-> GRASP gateway functions.

This document primarily defines how to perform service discovery across such a GRASP domain leveraging GRASP’s options to perform unsolicited flooding of announcements or flooding of requests, and finding the closest service instances. Also, the document allows for automatically discovering DNS-SD servers. Such features is meant to optimize the flooding traffic in some deployments.

The initial use case of this document is to support what in DNS-SD is done via mDNS but in larger networks - GRASP-Domains. Beside the efficient flooding, GRASP provides reliability and security, which are depending on the so called substrate used by GRASP for security and hop-by-hop/end-to-end transport, such as the Autonomic control plane (ACP), [RFC8994]. Providing compatibility with existing mDNS service announcer or clients is possible, but not described in this version of the document.

The encoding of information chosen in this document does not try to use GRASP solely as a transport layer, but to also leverage the CBOR structure of GRASP messages to natively encode the message elements required for services in a way that is most simple - instead of using GRASP only as, e.g., an encapsulation of otherwise unchanged DNS message encodings. This is done to minimize the amount of coding required (and not require any DNS code unless future gateway functions are required), to increase the simplicity, minimize the amount of data on the wire, and allow easier extensibility. On the downside, the mechanisms provided here do not cover the whole slew of possible options of DNS/DNS-SD, but instead only those deemed to be required. Others can be added later.
In support of service discovery, this document also defines name
discovery and schemes for reusable elements in GRASP objectives which
are designed to be extensible so that future work that identifies
elements required across multiple objectives do not need to define a
scheme how to do this.

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 makes use of terms and concepts defined in [RFC8990].

3. Specification

3.1. Service and Name Objectives

Unsolicited, flooded announcements (M_FLOOD) in GRASP and solicited
flooded discovery (M_DISCOVERY) operate on the unit of GRASP
technical objectives (identified by ‘objective-names’ as discussed in
Section 2.10 of [RFC8990]). Therefore, a scheme is required to
indicate services via ‘objective-names’.

Note: Future work may want to reuse the encodings related to
services (defined below in this document) inside other (multicast
or unicast only) objective exchanges, in which case the service
names are not impacted.

When a technical objective (simply referred to as objective) is meant
to be solely about a service name, the objective MUST uses an
‘objective-name’ of ‘SRV.<service-name>’. This naming scheme is
meant to avoid creating duplicates and, potentially, inconsistent
name registrations for those objectives vs. registrations done, for
example, for DNS-SD.

When an objective is meant announcement and discovery of a DNS
compatible <name> such as "www-internal" in "www-
internal.example.com", the objective SHOULD use an objective-name of
NAME.<name>. See Section 3.3.3 for more details.
3.2. Objective Value Reuseable Elements Structure

Because service discovery, as explained in the prior section, needs to utilize different objectives, it requires cross-objective standardized encoding of the elements of services. GRASP does not define standardized message elements for the message body (called "objective-value") of GRASP messages. Therefore, this document introduces such a feature.

```plaintext
objective-value /= { 1*elements }
elements       //== ( @rfcXXXX: { 1*relement } )

relement = ( relement-codepoint => relement-value )
relement-codepoint = uint
relement-value     = any
```

If an objective relies upon reusable elements, the 'objective-value' MUST be a CBOR map and the reusable elements are found under the key "@rfcXXXX".

Objectives that do not want reusable elements may use any objective-value format including a CBOR map, but they can not use the "@rfcXXXX" key if they use a map. This approach was chosen as the hopefully least intrusive mechanism given how by nature all of "objective-value" is meant to be defined by individual objective definitions.

The value of "@rfcXXXX" is a map of reusable elements. Each 'relement' has an IANA registered element-name and codepoint (see Section 6). The element-name is for documentation purposes only, CBOR encodings only use the numeric codepoint for encoding efficiency to minimize the risk for this solution to not be applicable to low-bitrate networks such as in IoT.

Format and semantic of the relement-value is determined by the specification of the reusable element as is the fact whether more than one instances of the same reusable element are permitted.

Reusable elements should be defined to be extensible. The methods used depend on the complexity of the element and the likely need to extend/modify the element with backward or non-backward compatible information. The following is a set of initial options to choose from:

Element values that are a map MUST permit and reserve key value 0 (numerical) for private extensions of the element defined by the individual objective.
Element values that are a map MUST NOT use bareword key values starting with a "_". These too are for private extensions defined by the individual objective.

Element values SHOULD be defined so that additional keys in maps and additional elements at the end of arrays can be ignored by prior versions of the definition. Whenever a newer definition is made for an element where this rule is violated, the element SHOULD be changed in a way for older version recipients to recognize that it is not compatible with it.

One method to indicate compatibility is a traditional version "<mayor>.<minor>". Within the same <mayor> version number, increasing <minor> version numbers must be backward compatible. Different <mayor> version numbers are not expected to be compatible with each other. If they are, then this can be indicated by including multiple version numbers.

A compressed form of version compatibility information is the use of a simple bitmask element where each bit indicates a version that the represented data is compatible with.

3.3. Reuseable Elements

3.3.1. Sender Loop Count

relement-codepoint //= ( &(sender-loop-count:1) => 1..255 )

Sender-loop-count is set by the sender of an objective message to the same value as the loop-count of the message. On receipt, distance = ( sender-loop-count - loop-count ) is the distance of the sender from the receiver in hops. This element can be used for informational purposes in M_FLOOD and M_DISCOVERY messages and may be required to be used in these messages by the specification of other elements (such as the service element described below). This element MUST occur at most once. If a receiver expects to use the distance but sender-loop-count was not announced, then distance SHOULD be assumed to be 255 by the receiver.

3.3.2. Service Element

The srv-element (service element) is a reusable element to request or announce a service instance or to request and list service instance names.
relement-codepoint // = ( & (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),
}

clocator = [ context, locator-option ]
context = cstr
locator-option = ; from GRASP

msg-type = & ( describe: 0, describe-request:1,
       enumerate:2, enumerate-request:3 )

Service: A service name registered according to RFC6335. If it is
not present, then objective-name MUST be SRV.<service-name> where
<service-name> is the service-name.

Instance: The <Instance> of a DNS-SD Service Instance Name ( 
          <Instance>.<Service>.<Domain>). It is optional, see
          Section 4.2.

Domain: The equivalent of the <Domain> field of a DNS-SD Service
Instance Name. If domain is not present, this is equivalent to
".local" in DNS (as introduced by mDNS) and implies the unnamed
"local" domain, which is the GRASP domain across which the message
is transmitted.

Priority, Weight: Service Instance selection criteria as defined in
RFC2782. If either one is not present, its value defaults to 0.

Kvpairs: Map of key/value pairs that are service parameters in the
same format as the key/value pairs in TXT field(s) of DNS-SD TXT
records as defined in RFC6763, section 6.3.

Range: Allows to flexibly combine distance and priority/weight based
service selection according to the definition of distance in
Section 3.3.1.

If min-distance is the distance of the closest service announcer,
and min-range the range announced by it, then the recipient MUST consider the priority/weight of all service announcers that are not further away than (min-distance + min-range). If not included, range defaults to 255.

If range is announced, the sender-loop-count element MUST also be announced.

Clocator: The "contextual locator" allows to indicate zero or more locators for the indicated service instance. The context element indicates in which context the locator-option is to be resolved. The reserved context value of "" (empty string) indicates the GRASP domain used, aka: the "local" context in which the service announcement is made. The reserved context value of "0" indicates the default routing context of the announcing node. This is often called "global table", "VRF 0" or "default VRF" on nodes using the "VRF" abstraction. Any other value is a string specifying a context such as another VRF.

The mechanism by which originator and recipient of the srv-element agree on common naming for contexts is outside the scope of this specification. The context therefore allows to indicate locators both for the context through which the GRASP message distributed the srv-element (GRASP domain) as well as that for other contexts. Assume the GRASP domain is the ACP, then clocators in ACP would have a context of "", clocators in the global routing table (part of the data-plane) a context of "0", and clocators on other VRFs (also part of data-plane) a clocator that is their string name.

If no locators are indicated, then the locator of the service(s) is the optional locator-option of the GRASP message in which the objective is contained meant to be used for the service(s) indicated and the clocator implied is "".

If locator(s) are indicated, the messages location-option must be ignored for the service (but may be necessary to be present for other purposes of the objective).

Msg-type Type (aka: intention) of the srv-element. If not present, it is assumed to be "describe".

Describe: Describes one service instance. At least one clocator is required for a positive response, all other fields are permitted, but optional. "Describe" is used in M_FLOOD for unsolicited announcements of services (flooded), in M_RESPONSE messages for solicited announcements of a service and in M_NEGOTIATE for negotiated announcements (both unicasted). If clocator is not included, then all fields except service and instance (and msg-
type and private) must not be included and the srv-element provides a negative reply: No information about this service/service instance. This is only permitted in unicast "describe" messages.

Describe-request: Request for a "describe" reply. It is used in M_DISCOVERY (flooded) for solicited discovery of services or in M_REQ_SYN (unicasted) for negotiated discovery of service instance(s). In "describe-request", only service is mandatory (but can be provided via the objective-name field of the message), and domain is optional. "Instance" is optional. If provided, then the recipient is asked to provide information about the named instance only. All other fields of srv-element are to be ignored by the receiver in this specification, but a semantic for setting them may be introduced in follow-up work, specifically to filter replies by the indicated fields.

"Describe-request" without instance MAY be answered by "Enumerate" (see below) if the responder has so many instances that it thinks the initiator should rather first select one or fewer instances and ask for their description. The sender of "Describe-request" MUST be prepared to accept that answer and as necessary follow up with "Describe-request" with the instance names of interest.

Enumerate: Used in the same GRASP messages as "describe", but instead of providing information about one service instance, it is listing service instance names. The purpose of enumerate is the same as browsing a service in DNS-SD. It would be followed by some human or automated selection of one or more instances and then a "describe" M_REQ_SYN request for those instances sent to the source of the "enumerate" to learn about the locators and other parameters of the service instances.

In this specification, all fields other than service, instance and domain (and msg-type and private) must be unset in "enumerate".

Enumerate-request: Requests an "enumerate" reply. It is used in the same way as "Describe-request" except that instance would usually not be set (because in that case it is more useful to send a "Describe-request").

3.3.3. Name Element

The NAME,<name> elements is meant to provide basic name resolution comparable to mDNS name resolution for GRASP domains where this is desirable and no better name resolution exist - for example in the ACP where there is no requirement for DNS.
Because the GRASP service lookup (unlike) DNS does not mandate that nodes have names (not even service instance names), the use of names is primarily meant to support legacy software. New designs should instead look up only services and service instance names, and nodes should announce their names as service instance names for the services they offer:

For example consider a GRASP (ACP) domain of "example.com". The node providing some "www" service could have a name "www-internal" which means GRASP objective NAME.www-internal, that objective value would include primarily the nodes IP address(es) and the port number for the www service would have to be guessed (80). Better, the node would announce GRASP objective SRV.www and the objective value would include the service instance name www-internal and the (TCP) port information (80 or a non-default port).

relement-codepoint //= ( & (name-element:3) => context-element )

context-element //= {
  *( & name:10 ) => tstr),
}

ipv6-address-option = [O_IPv4_ADDRESS, ipv6-address]
ipv4-address-option = [O_IPv6_ADDRESS, ipv6-address]
locator-option /= ipv4-address-option
locator-option /= ipv6-address-option

Name information is carried in the name-element relement. It is a context-element like the one used for srv-element except that it adds the name component and that it does not permit the service and instance components and that it allows only describe and describe-request values in the msg-type. Locators MUST use the ipv6-address-option or ipv4-address-option in the locator-option component.

TBD: Unclear if/how we should best formalize the differences in the context element permitted information between services and names. The above is quite informal.

Priority, weight, kvpairs, range (and of course private) MAY be used in describe messages to support multiple instances of the same name, as used for name anycast/prioritycast.
Nodes may have multiple names. These can be listed in the name component. If a node's names have the notion of a primary name and secondary names then the primary name should be the first in the list of names. In DNS-SD, the name pointed to by CNAME RRs can be considered to be the primary name. A describe-request for a non-primary name SHOULD return in the list of names the requested name and the primary name.

Note that there is no reverse lookup defined in this version of the document (no lookup from IP address to name).

4. Theory of Operation

4.1. Using GRASP Service Announcements

TBD: This section contains a range of details that should become normative in later versions.

This section provides a step by step walk-through of how to use GRASP service announcements and compares it to DNS-SD.

The most simple method to use GRASP service discovery is to select (and if still necessary, register) a <service-name> and start one or more agents (e.g.: ASAs) announcing their service instance(s) via GRASP. At minimum, an agent should periodically (default 60 seconds) announce the service instance via GRASP M_FLOOD messages as an objective SRV.<service-name> with a srv-element and a sender-loop-count element (default 255). The ttl of the GRASP message should be 3.5 times the announcement period, e.g.: 210000 msec.

Consumers of the service will use GRASP to learn of the service instances and select one. This approach is most similar to the use of DNS-SD with mDNS except that the scope of the announcement is a whole GRASP domain (such as the ACP) as opposed to a single IP subnet in mDNS and that mDNS primarily relies on request & reply but in its standard not on periodic unsolicited announcements. We describe here the unsolicited flooding option via M_FLOOD first because it is recommended for services with a dense population of service consumers and it is most simple to describe.

On the service announcer, the parameters priority, weight and range of the service instance can be selected from intent or configuration – or left at default. The default range 255 will result in selection of a random target of the service like in DNS-SD. Setting priority/weight allows to prioritize and weigh the selection as in DNS-SD. Setting range to 0 allows to select the closest target, priority/weight are only compared between targets of the same shortest
distance. Distance based options are not available in DNS-SD because it does not expect that network distance is available to arbitrary DNS-SD client. It is available to GRASP clients though. Using 0 < range < 255 allows for a hybrid priority/weight and distance based service selection (e.g.: Select the highest priority instance within a range of 5 hops).

If the service is a non-GRASP service, then the result of the service discovery has to be a transport locator to which the client can open a connection and talk the protocol implied by the service. This transport locator(s) have to be put into the clocator parameter. The context of the clocator would normally be "", aka: the transport locator is in the IP reachability associated with the GRASP domain (e.g.: IPv6 of the ACP for ACP GRASP domain).

If an ACP service is announced via ACP GRASP, then the locator(s) can be O_IPv6_LOCATOR or O_FQDN_LOCATOR. The O_IPv6_LOCATOR is used if the service is defined to be available via some transport layer port (TCP, UDP or other). The determination of the actual transport connection to be used is the same as in DNS-SD: If the transport protocol is not TCP or UDP, it has to be implied by the specification of <service-name> or can be detailed in kvpairs which carries the same information as DNS-TXT TXT RRs of the service. Alternatively, the transport-proto field of the locator can contain any valid IP protocol directly (TBD), which is not possible in DNS-SD.

Like DNS-SD, service discovery via GRASP does not require allocation and use of well-known ports for services. Unlike DNS-SD, there is no need in GRASP to define service instance names or target names. In DNS SD, PTR RRs resolve from a service name to a set of service instance named. SRV and TXT RRs resolve from service instance names to service instance parameters including the target. A target is the DNS host name of the service instance. It gets resolved via A/AAAA RRs to IPv4/IPv6 addresses of the target. In GRASP service discovery, host names are not used. Service instance names are optional too. Service instance names are useful for human diagnostics and human selection of service instances. In fully automated environments, they can be are less important. For diagnostic purposes, it is recommended to give service instances service instance names in GRASP service announcements.

A locator with O_URI_LOCATOR type can be used in GRASP to indicate a URI for the transport method for a service instance. If the URI includes a host part, care must be taken to use only IP addresses in the host part if the context of the GRASP domain does not support host name resolution - such as the ACP - or to use the GRASP name resolution mechanisms described elsewhere in this document. And that the addresses indicated are also reachable in the GRASP domain. For
example, in service announcements across a DULL GRASP domain, only the IPv6 link-local addresses on that subnet must be used (this applies equally when using the O_IPV6_LOCATOR).

Instead of using M_FLOOD to periodically announce service instances, M_DISCOVERY can be used to actively query for service instances. The msg-type type must then be "describe-request". Because no periodic flooding is necessary, this solution is more lightweight for the network when the number of requesting clients is small. Note though that the M_DISCOVERY will terminate as soon as a provider of the objective is found, so the service instances found will be based on distance and therefore selection of instance by priority and weight will not work equally well as with M_FLOOD. Consider for example a central service instance in the NOC that should always be used (for example for centralized operational diagnostics) unless the WAN connection is broken, in which case distributed backup service instances should be used. With the current logic of M_DISCOVERY this is not possible.

4.2. Further Comparison with DNS-SD

Neither the GRASP SRV.* objective-name, the service name nor any other parameter explicitly indicate the second label "_tcp" or "_udp" of DNS-SD entries. DNS-SD, RFC6763 explains how this is an unnecessary, historic artifact.

This version of the document does not define an equivalent to "_sub" structuring of service enumeration.

This version of the document does not define mechanisms for reverse resolution of arbitrary services: An inquirer may unicast M_SYNC_REC to a node with a series of objectives with specific service names of interest and describe-request, but there is no indication of "ANY" service.

4.3. Open Issues

TBD: Examine limitations mentioned in "in this version of the text/document".

TBD: The GRASP specification does currently only permit TCP and UDP for the transport-proto element. This draft should expand the GRASP definitions to permit any valid IP protocol. We just need to decide whether this should only apply to the locator in the srv element or also retroactive to the locator-option in GRASP messages (maybe not there?).
TBD: A fitting CBOR representation for a kvpair key without value needs to be specified so that it can be distinguished from an empty value as outlined in RFC6763 section 6.4.

TBD: In this version, every service/service-instance is an element by itself. Future versions of this document may add more encoding options to allow more compact encoding of recurring fields.

TBD: Is there a way in CDDL to formally define the string names of the relement-codepoint’s?

5. Security Considerations

TBD.

GRASP-related security issues are discussed in Section 3 of [RFC8990].

6. IANA Considerations

This document requests IANA to create a new "GRASP Objective Value Standard Elements" subregistry under the "GeneRic Autonomic Signaling Protocol (GRASP) Parameters" registry.

The values in this table are names and a unique numerical value assigned to each name. Future values MUST be assigned using the RFC Required policy as defined in Section 4.7 of [RFC8126]. The numerical value is simply to be assigned sequentially. The following initial values are assigned by this document:

sender-loop-count 1 [defined in rfcXXXX]

srv-element 2 [defined in rfcXXXX]

name-element 3 [defined in rfcXXXX]

This document updates the handling of the "GRASP Objective Names" Table introduced in the GRASP IANA considerations as follows:

Assignments for objective-names of the form "SRV.<text>" and "NAME.<text>" are special.

Assignment of "SRV.<text>" can only be requested if <text> is also a registered service-name according to RFC6335. The specification required for registration of a "GRASP Objective Name" MUST declare that the intended use of the objective name in GRASP is intended to be compatible with the indented use of the registered service name.
Registration of "SRV.<text>" in the "GRASP Objective Name" table is optional, but recommended for all new service-names that are meant to be used with GRASP. Non-registration can for example happen with DNS-SD <-> GRASP gateways that inject pre-existing service-names into GRASP. Note that according to the GRASP RFC, registration is mandatory, so this exemption for "SRV.<text>" is also an update to that specification.

There MUST NOT be any assignment for objective names of the form "NAME.<text>". These names are simply used by GRASP nodes without registration (just like names in mDNS).

7. Acknowledgements

8. Contributors

   Brian Carpenter

9. Change log [RFC Editor: Please remove]

9.1. 03 - Refresh

9.2. 02 - Revived after charter round 1 finished

   Reviving after ANIMA charter 01 is finished, adding new co-authors, contributors.

   Textual improvements, updating references.

9.3. 01 -

   Only refreshing, no changes since -00.

9.4. 00 - Initial version

10. References

10.1. Normative References


10.2. Informative References


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
35000 Rennes
France
Email: christian.jacquenet@orange.com

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

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.
1. Introduction

1.1. Overview

This document defines a protocol to support the autoconfiguration of Autonomic Control Plane (ACP) 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 BRISKI ([RFC8995]). BRISKI bootstrap is an instance of such a zero-touch bootstrap requiring auto-enablement of autoconfiguration after zero-touch 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 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.

* 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].

```plaintext
[M_FLOOD, 12340815, h’fd89b714f3db000020000064000001’, 210000,
{ 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’fd89b714f3db000020000064000001’, 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 adn 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. Some 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 choosen 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

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 decode causality. This may require network clock synchronization in the order of milliseconds, something which is easily achievable in todays 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.g., 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 used 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 the 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]

6.2. Informative References

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
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.
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 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 on 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.
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).
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/p10>;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 /getcertregtemplate 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]

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

[IEEE.802.1AR_2009]


9.2. Informative References


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 IDevID 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
Abstract

This document specifies the behavior 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 25 November 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
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 organization’s 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.
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.


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 an EST [RFC7030] 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.

EST [RFC7030] is not clear on how the CSR Attributes response should be structured, and in particular is not clear on how a server can instruct a client to include specific attribute values in its CSR. [I-D.richardson-lamps-rfc7030-csrattrs] clarifies how a server can use CSR Attributes response to specify specific values for attributes that the client should include in its CSR.

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 certificateAuthorities 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/MA SA using artifacts created during the manufacturing process of the Pledge. In step 2, the Pledge sends a voucher request to the Cloud RA/MA SA, and in response the Pledge receives an [RFC8366] format voucher from the Cloud RA/MA SA 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
  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/>"
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.";
}

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"}  
Registrator 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 following registration is requested:
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 cannot 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 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


8.2. Informative References

[I-D.richardson-lamps-rfc7030-csattrs]

[I-D.richardson-t2trg-idevid-considerations]

[IEEE802.1AR]

[RFC1918]

[RFC3688]

[RFC6020]

[RFC6125]


Authors’ Addresses

Owen Friel
Cisco
Email: ofriel@cisco.com

Rifaat Shekh-Yusef
Auth0
Email: rifaat.s.ietf@gmail.com

Michael Richardson
Sandelman Software Works
Email: mcr+ietf@sandelman.ca
Abstract

This document defines enhancements to bootstrapping a remote secure key infrastructure (BRSKI, [RFC8995]) to facilitate bootstrapping in domains featuring no or only timely limited connectivity between a pledge and the domain registrar. It specifically targets situations, in which the interaction model changes from a pledge-initiator-mode, as used in BRSKI, to a pledge-responder-mode as described in this document. To support both, BRSKI-PRM introduces a new registrar-agent component, which facilitates the communication between pledge and registrar during the bootstrapping phase. For the establishment of a trust relation between pledge and domain registrar, BRSKI-PRM relies on the exchange of authenticated self-contained objects (signature-wrapped objects). The defined approach is agnostic regarding the utilized enrollment protocol, deployed by the domain registrar to communicate with the Domain CA.
# Table of Contents

1. Introduction ................................................. 3
2. Terminology ................................................ 5
3. Scope of Solution .......................................... 6
   3.1. Supported Environments and Use Case Examples .......... 6
       3.1.1. Building Automation ............................. 7
       3.1.2. Infrastructure Isolation Policy ................. 7
       3.1.3. Less Operational Security in the Target-Domain ... 7
3.2. Limitations .............................................. 8
4. Requirements Discussion and Mapping to Solution-Elements ... 8
5. Architectural Overview and Communication Exchanges ........ 9
   5.1. Pledge-responder-mode (PRM): Registrar-agent Communication with Pledges .................. 10
   5.2. Agent-Proximity Assertion .............................. 13
   5.3. Behavior of Pledge in Pledge-Responder-Mode .......... 14
   5.4. Behavior of Registrar-Agent .......................... 15
       5.4.1. Discovery of Registrar by Registrar-Agent ....... 16
       5.4.2. Discovery of Pledge by Registrar-Agent ........ 16
   5.5. Bootstrapping Objects and Corresponding Exchanges .... 17
       5.5.1. Request Objects Acquisition by Registrar-Agent from Pledge ..................... 20
       5.5.2. Request Processing by the Registrar-Agent .......... 28
       5.5.3. Response Object Supply by Registrar-Agent to Pledge ..................... 39
       5.5.4. Telemetry status handling (registrar-agent - domain registrar) .................. 43
6. Artifacts .................................................. 45
   6.1. Voucher Request Artifact .............................. 45
       6.1.1. Tree Diagram .................................. 46
       6.1.2. YANG Module .................................. 46
7. IANA Considerations ........................................ 50
   7.1. BRSKI .well-known Registry ........................... 50
8. Privacy Considerations .................................... 50
1. Introduction

BRSKI as defined in [RFC8995] specifies a solution for secure zero-touch (automated) bootstrapping of devices (pledges) in a (customer) site domain. This includes the discovery of network elements in the customer site/domain and the exchange of security information necessary to establish trust between a pledge and the domain.

Security information about the customer site/domain, specifically the customer site/domain certificate, is exchanged utilizing voucher objects as defined in [RFC8366]. These vouchers are signed objects, provided via the domain registrar to the pledge and originate from a Manufacturer’s Authorized Signing Authority (MASA).

BRSKI addresses scenarios in which the pledge acts as client for the bootstrapping and is the initiator of the bootstrapping (this document refers to the approach as pledge-initiator-mode). In industrial environments the pledge may behave as a server and thus does not initiate the bootstrapping with the domain registrar. In this scenarios it is expected that the pledge will be triggered to generate request objects to be bootstrapped in the customer site/domain (this document refers to the approach as pledge-responder-mode). For this, an additional component is introduced acting as an agent for the domain registrar (registrar-agent) towards the pledge. This may be a functionality of a commissioning or configuration tool or it may be even co-located with the registrar.

In contrast to BRSKI the registrar-agent facilitates the object exchange with the pledge and provides/retrieves data objects to/from the domain registrar. For the interaction with the domain registrar the registrar-agent will use existing BRSKI [RFC8995] endpoints.
The term endpoint used in the context of this document is similar to resources in CoAP [RFC7252] and also in HTTP [RFC9110]. It is not used to describe a device. Endpoints are accessible via .well-known URIs.

The goal is to enhance BRSKI to support pledges in responder mode. This is addressed by

* introducing the registrar-agent as new component to facilitate the communication between the pledge and the registrar, if the pledge is in responder mode (acting as server).

* handling the security on application layer only to enable application of arbitrary transport means between the pledge and the domain registrar, by keeping the registrar-agent in the communication path. Examples may be connectivity via IP based networks (wired or wireless) but also connectivity via Bluetooth or NFC between the pledge and the registrar-agent.

* allowing to utilize credentials different from the pledge’s IDevID to establish a TLS connection to the domain registrar, which is necessary in case of using a registrar-agent.

* defining the interaction (data exchange and data objects) between a pledge acting as server and a registrar-agent and the domain registrar.

For the enrollment of devices BRSKI relies on EST [RFC7030] to request and distribute customer site/domain specific device certificates. EST in turn relies on a binding of the certification request to an underlying TLS connection between the EST client and the EST server. According to BRSKI the domain registrar acts as EST server and is also acting as registration authority (RA) for its domain. To utilize the EST server endpoints on the domain-registrar, the registrar-agent defined in this document will act as client towards the domain registrar. The registrar-agent will also act as client when communicating with the pledge in responder mode. Here, TLS with server-side, certificate-based authentication is not directly applicable, as the pledge only possesses an IDevID certificate, which does not contain a subject alternative name (SAN) for the customer site/domain and does also not contain a TLS server flag. This is one reason for relying on higher layer security by using signature wrapped objects for the exchange between the pledge and the registrar agent. A further reason is the application on different transports, for which TLS may not be available, like Bluetooth or NFC. Instead of using TLS to provide secure transport between the pledge and the registrar-agent, BRSKI-PRM will rely on an additional wrapping signature of the enrollment request by the
pledge. For EST [RFC7030] the registrar then needs to do additional pre-processing by verifying this signature, which is not present in EST.

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], section 1.2. The following terms are defined additionally:

authenticated self-contained object: Describes an object, which is cryptographically bound to the end entity (EE) certificate (IDevID certificate or LDEVID certificate). The binding is assumed to be provided through a digital signature of the actual object using the corresponding private key of the EE certificate.

CA: Certification authority, issues certificates.

Commissioning tool: Tool to interact with devices to provide configuration data

EE: End entity

mTLS: Mutual authenticated Transport Layer Security.

on-site: Describes a component or service or functionality available in the customer site/domain.

off-site: Describes a component or service or functionality not available in the customer site/domain. This may be a central site or a cloud service, to which only a temporary connection is available, or which is in a different administrative domain.

PER: Pledge-enrollment-request is an enrollment object signed by the pledge that requests to enroll in a domain

POP: Proof of possession (of a private key)

POI: Proof of identity

PVR: Pledge-voucher-request is a voucher request object signed by the pledge that requests to be part of a domain
RA: Registration authority, an optional system component to which a CA delegates certificate management functions such as authorization checks.

RER: Registrar-enrollment-request is a request object related to the PER send to the CA by the registrar.

RVR: Registrar-voucher-request is a request object containing the PVR sent to the MASA.

3. Scope of Solution

3.1. Supported Environments and Use Case Examples

BRSKI-PRM is applicable to environments where pledges may have different behavior: pledge-responder-mode, or pledges may have no direct connection to the domain registrar. Either way pledges are expected to be managed by the same registrar. This can be motivated by pledges deployed in environments not yet connected to the operational customer site/domain network, e.g., at construction time. Another environment relates to the assembly of cabinets, which are prepared in advance to be installed on a customer site/domain. As there is no direct connection to the registrar available in these environments the solution specified allows the pledges to act in a server role so they can be triggered for bootstrapping e.g., by a commissioning tool. As BRSKI focuses on the pledge in a client role, initiating the bootstrapping (pledge-initiator-mode), BRSKI-PRM defines pledges acting as a server (pledge-responder-mode) responding to requests for PVR and PER objects and consumption of the result objects.

The following examples motivate support of BRSKI-PRM to support pledges acting as server as well as pledges with limited connectivity to the registrar.

While BRSKI-PRM defines support for pledges in responder mode, there may be pledges, which can act in both modes, initiator and responder. In these cases BRSKI-PRM can be combined with BRSKI as defined in [RFC8995] or BRSKI-AE [I-D.ietf-anima-brski-ae] to allow for more bootstrapping flexibility.
3.1.1. Building Automation

In building automation a typical use case exists where a detached building (or a cabinet) or the basement of a building is equipped with sensors, actuators and controllers, but with only limited or no connection to the central building management system. This limited connectivity may exist during installation time or also during operation time. During the installation in the basement, a service technician collects the device specific information from the basement network and provides them to the central building management system, e.g., using a laptop or a mobile device to transport the information. A domain registrar may be part of the central building management system and already be operational in the installation network. The central building management system can then provide operational parameters for the specific devices in the basement. This operational parameters may comprise values and settings required in the operational phase of the sensors/actuators, among them a certificate issued by the operator to authenticate against other components and services. These operational parameters are then provided to the devices in the basement facilitated by the service technician’s laptop.

3.1.2. Infrastructure Isolation Policy

This refers to any case in which the 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 a domain registrar may be allowed in carefully controlled short periods of time, for example when a batch of new devices are deployed, but prohibited at other times.

3.1.3. Less Operational Security in the Target-Domain

The registration authority (RA) performing the authorization of a certificate request is a critical PKI component and therefore requires higher operational security than other components utilizing the issued certificates. CAs may also require higher security in the registration procedures. Especially the CA/Browser [CABF] forum increases the security requirements in the certificate issuance procedures for publicly trusted certificates. There may be situations in which the customer site/domain does not offer enough security to operate a RA/CA and therefore this service is transferred to a backend that offers a higher level of operational security.
3.2. Limitations

The mechanism described in this document presume the availability of the pledge to communicate with the registrar-agent. This may not be possible in constrained environments where, in particular, power must be conserved. In these situations, it is anticipated that the transceiver will be powered down most of the time. This presents a rendezvous problem: the pledge is unavailable for certain periods of time, and the registrar-agent is similarly presumed to be unavailable for certain periods of time.

4. Requirements Discussion and Mapping to Solution-Elements

Based on the intended target environment described in Section 3.1 and the application examples described in Section 3.1 the following requirements are derived to support bootstrapping of pledges in responder mode (acting as server).

* To facilitate the communication between a pledge in responder mode and registrar, additional functionality is needed either on the registrar (if the registrar needs to interact with pledge in responder mode directly) or as a stand-alone component. This component acts as an agent of the registrar to trigger the pledge to generate request objects for voucher and enrollment. These voucher an enrollment request objects are than to be provided by the so called registrar-agent to the registrar. This requires the definition of endpoints on the pledge.

* The communication between the registrar-agent and the pledge MUST not rely on transport layer security (TLS) to support also other technology stacks (e.g., BTLE). Therefore authenticated self-contained objects are required.

* The registrar-agent must be authenticated by the registrar as a component, acting on behalf of the registrar. In addition the registrar must be able to verify, which registrar-agent was in direct contact with the pledge.

* The pledge cannot get the assertion with value "proximity" in the voucher, as it was not in direct contact with the registrar for bootstrapping. Therefore the "agent-proximity" assertion value is necessary for distinguishing assertions the MASA can state.

At least the following properties are required for the voucher and enrollment objects:
* Proof of Identity (POI): provides data-origin authentication of a data object, e.g., a voucher request or an enrollment request, utilizing an existing IDevID. Certificate updates may utilize the certificate that is to be updated.

* Proof of Possession (POP): proves that an entity possesses and controls the private key corresponding to the public key contained in the certification request, typically by adding a signature using the private key to the enrollment request object.

Solution examples based on existing technology are provided with the focus on existing IETF RFCs:

* Voucher request and response objects as used in [RFC8995] already provide both, POP and POI, through a digital signature to protect the integrity of the voucher object, while the corresponding signing certificate contains the identity of the signer.

* Certification request objects: Certification requests are data structures containing the information from a requester for a CA to create a certificate. The certification request format in BRSKI is PKCS#10 [RFC2986]. In PKCS#10, the structure is signed to ensure integrity protection and proof of possession of the private key of the requester that corresponds to the contained public key. In the application examples, this POP alone is not sufficient. POI is also required for the certification request object and therefore needs to be additionally bound to the existing credential of the pledge (IDevID). This binding supports the authorization decision for the certification request through a proof of identity (POI). The binding of data origin authentication or POI to the certification request may be provided directly by the certification request object. While BRSKI uses the binding to TLS, BRSKI-PRM aims at an additional signature of the PKCS#10 object using existing credentials on the pledge (IDevID). This ensures independence of the selected transport.

5. Architectural Overview and Communication Exchanges

For BRSKI with pledge in responder mode, the base system architecture defined in BRSKI [RFC8995] is enhanced to facilitate the new use cases. The pledge-responder-mode allows delegated bootstrapping using a registrar-agent instead of a direct connection between the pledge and the domain registrar. The communication model between registrar-agent and pledge in this document assumes that the pledge is acting as server and responds to requests.
Necessary enhancements to support authenticated self-contained objects for certificate enrollment are kept at a minimum to enable reuse of already defined architecture elements and interactions.

For the authenticated self-contained objects used for the certification request, BRSKI-PRM relies on the defined message wrapping mechanisms of the enrollment protocols stated in Section 4 above.

The security used within the document for bootstrapping objects produced or consumed by the pledge bases on JOSE [RFC7515]. In constraint environments it may provided based on COSE [RFC8152].

An abstract overview of the BRSKI-PRM protocol can be found in [BRSKI-PRM-abstract].

5.1. Pledge-responder-mode (PRM): Registrar-agent Communication with Pledges

To support mutual trust establishment between the domain registrar and pledges not directly connected to the customer site/domain, this document specifies the exchange of authenticated self-contained objects (the voucher request/response objects as known from BRSKI and the enrollment request/response objects as introduced by BRSKI-PRM) with the help of a registrar-agent. This allows independence from protection provided by the utilized transport protocol.

The registrar-agent may be implemented as an integrated functionality of a commissioning tool or be co-located with the registrar itself. This leads to extensions of the logical components in the BRSKI architecture as shown in Figure 1. Note that the Join Proxy is neglected in the figure as not needed by the registrar-agent. The registrar-agent interacts with the pledge to transfer the required data objects for bootstrapping, which are then also exchanged between the registrar-agent and the domain registrar. The addition of the registrar-agent influences the sequences of the data exchange between the pledge and the domain registrar as described in [RFC8995]. To enable reuse of BRSKI defined functionality as much as possible, BRSKI-PRM uses existing endpoints were the required functionality is provided enhances existing with new supported media types, e.g., for JWS voucher defines new endpoints where additional functionality is required, e.g., for wrapped certification request.
For authentication to the domain registrar, the registrar-agent uses its LDevID(RegAgt). The provisioning of the registrar-agent LDevID is out of scope for this document, but may be done in advance using a separate BRSKI run or by other means like configuration. It is recommended to use short lived registrar-agent LDevIDs in the range of days or weeks as outlined in Section 9.3.

If a registrar detects a request that originates from a registrar-agent it is able to switch the operational mode from BRSKI to BRSKI-PRM. This may be supported by a specific naming in the SAN (subject alternative name) component of the LDevID(RegAgt) certificate. Alternatively, the domain may feature an own issuing CA for registrar-agent LDevID certificates. This allows the registrar to detect registrar-agents based on the issuing CA.

The following list describes the components in a (customer) site domain:

Figure 1: Architecture overview using registrar-agent
Pledge: The pledge is expected to respond with the necessary data objects for bootstrapping to the registrar-agent. The protocol used between the pledge and the registrar-agent is assumed to be HTTP in the context of this document. Other protocols may be used like CoAP, Bluetooth, or NFC, but are out of scope of this document. A pledge acting as a server during bootstrapping leads to some differences to BRSKI:

- Discovery of the pledge by the registrar-agent must be possible.

- As the registrar-agent must be able to request data objects for bootstrapping of the pledge, the pledge must offer corresponding endpoints.

- The registrar-agent may provide additional data to the pledge in the context of the voucher triggering request, to make itself visible to the domain registrar.

- Order of exchanges in the call flow may be different as the registrar-agent collects both objects, PVR objects and PER objects, at once and provides them to the registrar. This approach may also be used to perform a bulk bootstrapping of several devices.

- The data objects utilized for the data exchange between the pledge and the registrar are self-contained authenticated objects (signature-wrapped objects).

Registrar-agent: provides a communication path to exchange data objects between the pledge and the domain registrar. The registrar-agent brokers in situations, in which the domain registrar is not directly reachable by the pledge, either due to a different technology stack or due to missing connectivity. The registrar-agent triggers a pledge to create bootstrapping artifacts such as voucher-request objects and enrollment-request objects on one or multiple pledges and performs a (bulk) bootstrapping based on the collected data. The registrar-agent is expected to possess information of the domain registrar (i.e., LDevID(Reg) certificate, LDevID(CA) certificate, address), either by configuration or by using the discovery mechanism defined in [RFC8995]. There is no trust assumption between the pledge and the registrar-agent as only authenticated self-contained objects are used, which are transported via the registrar-agent and provided either by the pledge or the registrar. The trust assumption between the registrar-agent and the registrar is based on the LDevID of the registrar-agent, provided by the PKI responsible for the domain.
This allows the registrar-agent to authenticate towards the registrar, e.g., in a TLS handshake. Based on this, the registrar is able to distinguish a pledge from a registrar-agent during the session establishment and also to verify that the registrar-agent is authorized to perform the bootstrapping of the distinct pledge.

* Join Proxy (not shown): same functionality as described in [RFC8995] if needed. Note that a registrar-agent may use a join proxy to facilitate the TLS connection to the registrar, in the same way that a BRSKI pledge would use a join proxy. This is useful in cases where the registrar-agent does not have full IP connectivity via the domain network, or cases where it has no other means to locate the registrar on the network.

* Domain Registrar: In general the domain registrar fulfills the same functionality regarding the bootstrapping of the pledge in a (customer) site domain by facilitating the communication of the pledge with the MASA service and the domain PKI service. In contrast to [RFC8995], the domain registrar does not interact with a pledge directly but through the registrar-agent. The registrar detects if the bootstrapping is performed by the pledge directly or by the registrar-agent. The manufacturer provided components/services (MASA and Ownership tracker) are used as defined in [RFC8995]. For issuing a voucher, the MASA may perform additional checks on voucher-request objects, to issue a voucher indicating agent-proximity instead of (registrar-)proximity.

5.2. Agent-Proximity Assertion

"Agent-proximity" is a weaker assertion than "proximity". It is defined as additional assertion type in [I-D.ietf-anima-rfc8366bis] "agent-proximity" is a statement, that the proximity registrar certificate was provided via the registrar-agent as defined in Section 5.5 and not directly to the pledge. This can be verified by the registrar and also by the MASA during the voucher-request processing. Note that at the time of creating the voucher-request, the pledge cannot verify the registrar's LDevID(Reg) certificate and has no proof-of-possession of the corresponding private key for the certificate. The pledge therefore accepts the LDevID(Reg) provisionally until it receives the voucher as described in Section 5.5.3.

Trust handover to the domain is established via the "pinned-domain-certificate" in the voucher.

In contrast, "proximity" provides a statement, that the pledge was in direct contact with the registrar and was able to verify proof-of-possession of the private key in the context of the TLS handshake.
The provisionally accepted LDevID(Reg) certificate can be verified after the voucher has been processed by the pledge. As the returned voucher includes an additional signature by the registrar, the pledge can also verify that the registrar possesses the corresponding private key.

5.3. Behavior of Pledge in Pledge-Responder-Mode

In contrast to BRSKI the pledge acts as server. It is triggered by the registrar-agent for the generation of the PVR and PER objects as well as for the processing of the response objects and the generation of status information. Due to the use of the registrar-agent, the interaction with the domain registrar is changed as shown in Figure 4. To enable interaction with the registrar-agent, the pledge provides endpoints using the BRSKI defined endpoints based on the "/.well-known/brski" URI tree.

The following endpoints are defined for the _pledge_ in this document. The URI path begins with "http://www.example.com/.well-known/brski" followed by a path-suffix that indicates the intended operation.

Operations and their corresponding URIs:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operation path</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trigger pledge-voucher-request creation</td>
<td>/pledge-voucher-request</td>
<td>Section 5.5.1</td>
</tr>
<tr>
<td>Returns PVR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trigger pledge-enrollment-request</td>
<td>/pledge-enrollment-request</td>
<td>Section 5.5.1</td>
</tr>
<tr>
<td>Returns PER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provide voucher to pledge</td>
<td>/pledge-voucher</td>
<td>Section 5.5.3</td>
</tr>
<tr>
<td>Returns pledge-voucher-status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provide enrollment response to pledge</td>
<td>/pledge-enrollment</td>
<td>Section 5.5.3</td>
</tr>
<tr>
<td>Returns pledge-enrollment-status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Provide CA certs to pledge</td>
<td>/pledge-CACerts</td>
<td>Section 5.5.3</td>
</tr>
</tbody>
</table>


5.4. Behavior of Registrar-Agent

The registrar-agent is a new component in the BRSKI context. It provides connectivity between the pledge and the domain registrar and reuses the endpoints of the domain registrar side already specified in [RFC8995]. It facilitates the exchange of data objects between the pledge and the domain registrar, which are the voucher request/response objects, the enrollment request/response objects, as well as related status objects. For the communication with the pledge the registrar-agent utilizes communication endpoints provided by the pledge. The transport in this specification is based on HTTP but may also be done using other transport mechanisms. This new component changes the general interaction between the pledge and the domain registrar as shown in Figure 1.

The registrar-agent is expected to already possess an LDevID(RegAgt) to authenticate to the domain registrar. The registrar-agent will use this LDevID(RegAgt) when establishing the TLS session with the domain registrar for TLS client authentication. The LDevID(RegAgt) certificate MUST include a SubjectKeyIdentifier (SKID), which is used as reference in the context of an agent-signed-data object as defined in Section 5.5.1. Note that this is an additional requirement for issuing the certificate, as [IEEE-802.1AR] only requires the SKID to be included for intermediate CA certificates. In BRSKI-PRM, the SKID is used in favor of a certificate fingerprint to avoid additional computations.

Using an LDevID for TLS client authentication is a deviation from [RFC8995], in which the pledge’s IDevID credential is used to perform TLS client authentication. The use of the LDevID(RegAgt) allows the domain registrar to distinguish, if bootstrapping is initiated from a pledge or from a registrar-agent and adopt the internal handling accordingly. As BRSKI-PRM uses authenticated self-contained data objects between the pledge and the domain registrar, the binding of the pledge identity to the request object is provided by the data object signature employing the pledge’s IDevID. The objects exchanged between the pledge and the domain registrar used in the context of this specifications are JOSE objects.
In addition to the LDevID(RegAgt), the registrar-agent is provided with the product-serial-numbers of the pledges to be bootstrapped. This is necessary to allow the discovery of pledges by the registrar-agent using mDNS. The list may be provided by administrative means or the registrar agent may get the information via an interaction with the pledge. For instance, [RFC9238] describes scanning of a QR code, the product-serial-number would be initialized from the 12N B005 Product Serial Number.

According to [RFC8995] section 5.3, the domain registrar performs the pledge authorization for bootstrapping within his domain based on the pledge voucher-request object.

The following information must therefore be available at the registrar-agent:

* LDevID(RegAgt): own operational key pair.
* LDevID(reg) certificate: certificate of the domain registrar.
* Serial-number(s): product-serial-number(s) of pledge(s) to be bootstrapped.

5.4.1. Discovery of Registrar by Registrar-Agent

The discovery of the domain registrar may be done as specified in [RFC8995] with the deviation that it is done between the registrar-agent and the domain registrar. Alternatively, the registrar-agent may be configured with the address of the domain registrar and the certificate of the domain registrar.

5.4.2. Discovery of Pledge by Registrar-Agent

The discovery of the pledge by registrar-agent should be done by using DNS-based Service Discovery [RFC6763] over Multicast DNS [RFC6762] to discover the pledge. The pledge constructs a local host name based on device local information (product-serial-number), which results in "product-serial-number._brski-pledge._tcp.local".

The registrar-agent MAY use

* "product-serial-number._brski-pledge._tcp.local", to discover a specific pledge, e.g., when connected to a local network.
* ".brski-pledge._tcp.local" to get a list of pledges to be bootstrapped.
To be able to detect the pledge using mDNS, network connectivity is required. For Ethernet it is provided by simply connecting the network cable. For WIFI networks, connectivity can be provided by using a pre-agreed SSID for bootstrapping. The same approach can be used by 6LoWPAN/mesh using a pre-agreed PAN ID. How to gain network connectivity is out of scope of this document.

5.5. Bootstrapping Objects and Corresponding Exchanges

The interaction of the pledge with the registrar-agent may be accomplished using different transport means (protocols and or network technologies). For this document the usage of HTTP is targeted as in BRSKI. Alternatives may be CoAP, Bluetooth Low Energy (BLE), or Nearfield Communication (NFC). This requires independence of the exchanged data objects between the pledge and the registrar from transport security. These transport means may differ from, and are independent from, the ones used between the registrar-agent and the registrar. Therefore, authenticated self-contained objects (here: signature-wrapped objects) are applied in the data exchange between the pledge and the registrar.

The registrar-agent provides the domain-registrar certificate (LDevID(Reg) certificate) to the pledge to be included into the "agent-provided-proximity-registrar-certificate" leaf of the PVR object. This enables the registrar to verify, that it is the target registrar for handling the request. The registrar certificate may be configured at the registrar-agent or may be fetched by the registrar-agent based on a prior TLS connection establishment with the domain registrar. In addition, the registrar-agent provides agent-signed-data containing the product-serial-number in the body, signed with the LDevID(RegAgt). This enables the registrar to verify and log, which registrar-agent was in contact with the pledge, when verifying the PVR. Optionally the registrar-agent may provide its LDevID(RegAgt) certificate (and optionally also the issuing CA certificate) to the pledge to be used in the "agent-sign-cert" component of the PVR. If contained, the LDevID(RegAgt) certificate MUST be the first certificate in the array. Note, this may be omitted in constraint environments to save bandwidth between the registrar-agent and the pledge. If not contained, the registrar MUST fetch the LDevID(RegAgt) certificate based on the SubjectKeyIdentifier (SKID) in the header of the agent-signed-data of the PVR. The registrar includes the LDevID(RegAgt) certificate information into the RVR if the PVRs contains the assertion of "agent-proximity".

The MASA in turn verifies the LDevID(Reg) certificate is included in the PVR (prior-signed-voucher-request) in the "agent-provided-proximity-registrar-certificate" leaf and may assert in the voucher
"verified" or "logged" instead of "proximity", as there is no direct connection between the pledge and the registrar. In addition, the MASA can provide the assertion "agent-proximity" as following. If the LDevID(RegAgt) certificate information is contained in the "agent-sign-cert" component of the RVR, the MASA can verify the signature of the agent-signed-data contained in the prior-signed-voucher-request. If both can be verified successfully, the MASA can assert "agent-proximity" in the voucher. Otherwise, it may assert "verified" or "logged". Depending on the MASA verification policy, it may also respond with a suitable 4xx or 5xx error HTTP response code as described in section 5.6 of [RFC8995]. The voucher can then be supplied via the registrar to the registrar-agent.

Figure 3 provides an overview of the exchanges detailed in the following sub sections.

<table>
<thead>
<tr>
<th>Pledge</th>
<th>Registrar</th>
<th>Domain Registrar</th>
<th>Domain CA</th>
<th>Vendor Service (MASA)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Internet</td>
</tr>
</tbody>
</table>

/* discover pledge */

mDNS query

<-------------

------------->

/* trigger PVR and PER generation */

<- vTrigger --
- Voucher-Req->

<- eTrigger --
- Enroll-Req->

/* provide PVR to infrastructure */

<------ TLS ------|

[Reg-Agt authenticated and authorized?]

-- Voucher-Req -->
[Reg-Agt authorized?]
[accept device?]
[contact vendor]

-------- Voucher-Req -------->
[extract DomainID]
[update audit log]

<------ Voucher ----->
/* provide PER to infrastructure */

-- Enroll-Req -->
- Cert-Req -->
<-Certificate-

<- Enroll-Resp --

/* query cACerts from infrastructure */

-- cACerts-Req -->
<- cACerts-Resp --

/* provide voucher and certificate and collect status info */

<- Voucher --
-- vStatus -->
<- cACerts --
<- Enroll-Resp-
-- eStatus -->

/* provide voucher status and enroll status to registrar */

<------ TLS ------>
---- vStatus ---->
-- req. device audit log -->
<----- device audit log ---->
[verify audit log]

---- eStatus ---->

Figure 3: Overview pledge-responder-mode exchanges

The following sub sections split the interactions between the
different components into:

* Section 5.5.1 describes objects exchanged between the registrar-
  agent and the pledge.

* Section 5.5.2 describes objects exchanged between the registrar-
  agent and the registrar and also the interaction of the registrar
  with the MASA and the domain CA.

* Section 5.5.3 describes objects exchanged between the registrar-
  agent and the pledge including the status objects.

* Section 5.5.4 describes the status handling addresses the
  exchanges between the registrar-agent and the registrar.
5.5.1. Request Objects Acquisition by Registrar-Agent from Pledge

The following description assumes that the registrar-agent already discovered the pledge. This may be done as described in Section 5.4.2 based on mDNS.

The focus is on the exchange of signature-wrapped objects using endpoints defined for the pledge in Section 5.3.

Preconditions:

* Pledge: possesses IDevID

* Registrar-agent: possesses/trusts IDevID CA certificate and an own LDevID(RegAgt) EE credential for the registrar domain. In addition, the registrar-agent MUST know the product-serial-number(s) of the pledge(s) to be bootstrapped. The registrar-agent MAY be provided with the product-serial-number in different ways:
  - configured, e.g., as a list of pledges to be bootstrapped via QR code scanning
  - discovered by using standard approaches like mDNS as described in Section 5.4.2
  - discovered by using a vendor specific approach, e.g., RF beacons

* Registrar: possesses/trusts IDevID CA certificate and an own LDevID(Reg) credential.
Triggering the pledge to create the PVR is done using HTTP POST on the defined pledge endpoint "/.well-known/brski/pledge-voucher-request".

The registrar-agent PVR Content-Type header is: application/json. It defines a JSON document to provide three parameters:

* agent-provided-proximity-registrar-cert: base64-encoded LDevID(Reg) TLS certificate.
* agent-signed-data: base64-encoded JWS-object.
* agent-sign-cert: array of base64-encoded certificate data (optional).

The trigger for the pledge to create a PVR is depicted in the following figure:
The pledge provisionally accepts the agent-provided-proximity-registrar-cert and can verify it once it has received the voucher. If the optionally agent-sign-cert data is included the pledge MAY verify at least the signature of the agent-signed-data using the first contained certificate, which is the LDevID(RegAgt) certificate. If further certificates are contained in the agent-sign-cert, they enable also the certificate chain validation. The pledge may not verify the agent-sign-cert itself as the domain trust has not been established at this point of the communication. It can be done, after the voucher has been received.

The agent-signed-data is a JOSE object and contains the following information:

The header of the agent-signed-data contains:

* alg: algorithm used for creating the object signature.
* kid: MUST contain the base64-encoded bytes of the SubjectKeyIdentifier (the "KeyIdentifier" OCTET STRING value), excluding the ASN.1 encoding of "OCTET STRING" of the LDevID(RegAgt) certificate.

The body of the agent-signed-data contains an ietf-voucher-request-prm:agent-signed-data element (defined in Section 6.1):

* created-on: MUST contain the creation date and time in yang:date-and-time format.
* serial-number: MUST contain the product-serial-number as type string as defined in [RFC8995], section 2.3.1. The serial-number corresponds with the product-serial-number contained in the X520SerialNumber field of the IDevID certificate of the pledge.
Figure 6: Representation of agent-signed-data

Upon receiving the voucher-request trigger, the pledge SHALL construct the body of the PVR object as defined in [RFC8995]. It will contain additional information provided by the registrar-agent as specified in the following. This object becomes a JSON-in-JWS object as defined in [I-D.ietf-anima-jws-voucher]. If the pledge is unable to construct the PVR it SHOULD respond with HTTP 406 error code to the registrar-agent to indicate that it is not able to create the PVR.

The header of the PVR SHALL contain the following parameters as defined in [RFC7515]:

* **alg**: algorithm used for creating the object signature.

* **x5c**: contains the base64-encoded pledge IDevID certificate. It may optionally contain the certificate chain for this certificate.

The payload of the PVR object MUST contain the following parameters as part of the ietf-voucher-request-prm:voucher as defined in [RFC8995]:

* **created-on**: SHALL contain the current date and time in yang:date-and-time format.

* **nonce**: SHALL contain a cryptographically strong random or pseudo-random number.
* serial-number: SHALL contain the pledge product-serial-number as X520SerialNumber.

* assertion: SHALL contain the requested voucher assertion "agent-proximity".

The ietf-voucher-request:voucher is enhanced with additional parameters:

* agent-provided-proximity-registrar-cert: MUST be included and contains the base64-encoded LDevID(Reg) certificate (provided as trigger parameter by the registrar-agent).

* agent-signed-data: MUST contain the base64-encoded agent-signed-data (as defined in Figure 6) and provided as trigger parameter.

* agent-sign-cert: MAY contain the certificate or certificate chain of the registrar-agent as array of base64-encoded certificate information. It starts from the base64-encoded LDevID(RegAgt) certificate optionally followed by the issuing CA certificate and potential further certificates. If supported, it MUST at least contain the LDevID(RegAgt) certificate provided as trigger parameter.

The enhancements of the YANG module for the ietf-voucher-request with these new leaves are defined in Section 6.1.

The object is signed using the pledge’s IDevID credential contained as x5c parameter of the JOSE header.
{  
  "payload": {  
    "ietf-voucher-request-prm:voucher": {  
      "created-on": "2021-04-16T00:00:02.000Z",  
      "nonce": "eDs++/FuDHGUuN3E14CQ==",  
      "serial-number": "callee4711",  
      "assertion": "agent-proximity",  
      "agent-provided-proximity-registrar-cert": "base64encodedvalue==",  
      "agent-signed-data": "base64encodedvalue==",  
      "agent-sign-cert": [  
        "base64encodedvalue==",  
        "base64encodedvalue==",  
        "..."  
      ],  
    },  
    "signatures": [  
      {  
        "protected": {  
          "alg": "ES256",  
          "x5c": [ "MIIB2jCC...dA==" ]  
        },  
        "signature": "base64encodedvalue=="  
      }  
    ]  
  }  
}  

Figure 7: Representation of PVR

The PVR Content-Type is defined in [I-D.ietf-anima-jws-voucher] as application/voucher-jws+json.

The pledge SHOULD include this Content-Type header field indicating the included media type for the voucher response. Note that this is also an indication regarding the acceptable format of the voucher response. This format is included by the registrar as described in Section 5.5.2.

Once the registrar-agent has received the PVR it can trigger the pledge to generate an enrollment-request object. As in BRSKI the enrollment request object is a PKCS#10, but additionally signed using the pledge’s IDevID. Note, as the initial enrollment aims to request a generic certificate, no certificate attributes are provided to the pledge.

Triggering the pledge to create the enrollment-request is done using HTTP POST on the defined pledge endpoint "/.well-known/brski/pledge-enrollment-request".

The registrar-agent PER Content-Type header is: application/json with an empty body. Note that using HTTP POST allows for an empty body, but also to provide additional data, like CSR attributes or information about the enroll type: "enroll-generic-cert" or "reenroll-generic-cert". The "enroll-generic-cert" case is shown in Figure 8.

```json
{
  "enroll-type" = "enroll-generic-cert"
}
```

Figure 8: Example of trigger to create a PER

In the following the enrollment is described as initial enrollment with an empty body.

Upon receiving the enrollment-trigger, the pledge SHALL construct the PER as authenticated self-contained object. The CSR already assures proof of possession of the private key corresponding to the contained public key. In addition, based on the additional signature using the IDevID, proof of identity is provided. Here, a JOSE object is being created in which the body utilizes the YANG module ietf-ztp-types with the grouping for csr-grouping for the CSR as defined in [I-D.ietf-netconf-sztp-csr].

Depending on the capability of the pledge, it constructs the enrollment request as plain PKCS#10. Note that the focus in this use case is placed on PKCS#10 as PKCS#10 can be transmitted in different enrollment protocols in the infrastructure like EST, CMP, CMS, and SCEP. If the pledge is already implementing an enrollment protocol, it may leverage that functionality for the creation of the enrollment request object. Note also that [I-D.ietf-netconf-sztp-csr] also allows for inclusion of certification request objects such as CMP or CMC.

The pledge SHOULD construct the PER as PKCS#10 object. In BRSKI-PRM it MUST sign it additionally with its IDevID credential to provide proof-of-identity bound to the PKCS#10 as described below.

If the pledge is unable to construct the enrollment-request it SHOULD respond with HTTP 406 error code to the registrar-agent to indicate that it is not able to create the enrollment-request.

A successful enrollment will result in a generic LDevID certificate for the pledge in the new domain, which can be used to request further (application specific) LDevID certificates if necessary for its operation. The registrar-agent SHALL use the endpoints specified in this document.
[I-D.ietf-netconf-sztp-csr] considers PKCS#10 but also CMP and CMC as certification request format. Note that the wrapping signature is only necessary for plain PKCS#10 as other request formats like CMP and CMS support the signature wrapping as part of their own certificate request format.

The registrar-agent enrollment-request Content-Type header for a wrapped PKCS#10 is: application/jose+json

The header of the pledge enrollment-request SHALL contain the following parameter as defined in [RFC7515]:

*  alg: algorithm used for creating the object signature.

*  x5c: contains the base64-encoded pledge IDevID certificate. It may optionally contain the certificate chain for this certificate.

The body of the pledge enrollment-request object SHOULD contain a P10 parameter (for PKCS#10) as defined for ietf-ztp-types:p10-csr in [I-D.ietf-netconf-sztp-csr]:

*  P10: contains the base64-encoded PKCS#10 of the pledge.

The JOSE object is signed using the pledge’s IDevID credential, which corresponds to the certificate signaled in the JOSE header.

```
{
    "payload": {
        "ietf-ztp-types": {
            "p10-csr": "base64encodedvalue="
        },
        "signatures": [
            {
                "protected": {
                    "alg": "ES256",
                    "x5c": [ "MIIB2jCC...dA=" ]
                },
                "signature": "base64encodedvalue="
            }
        ]
    }
}
```

Figure 9: Representation of PER

With the collected PVR object and the PER object, the registrar-agent starts the interaction with the domain registrar.
As the registrar-agent is intended to facilitate communication between the pledge and the domain registrar, a collection of requests from more than one pledge is possible, allowing a bulk bootstrapping of multiple pledges using the same connection between the registrar-agent and the domain registrar.

5.5.2. Request Processing by the Registrar-Agent

The BRSKI-PRM bootstrapping exchanges between registrar-agent and domain registrar resemble the BRSKI exchanges between pledge and domain registrar (pledge-initiator-mode) with some deviations.

Preconditions:

* Registrar-agent: possesses it’s own LDevID(RegAgt) credentials of the site domain. In addition, it may possess the IDevID CA certificate of the pledge vendor/manufacturer to verify the pledge certificate in the received request messages. It has the address of the domain registrar through configuration or by discovery, e.g., mDNS/DNSSD. The registrar-agent has acquired PVR and PER objects(s).

* Registrar: possesses the IDevID CA certificate of the pledge vendor/manufacturer and an it’s own LDevID(Reg) credentials of the site domain.

* MASA: possesses it’s own vendor/manufacturer credentials (voucher signing key, TLS server certificate) related to pledges IDevID and the site-specific LDevID CA certificate.
The registrar-agent establishes a TLS connection with the registrar. As already stated in [RFC8995], the use of TLS 1.3 (or newer) is encouraged. TLS 1.2 or newer is REQUIRED on the registrar-agent side. TLS 1.3 (or newer) SHOULD be available on the registrar, but TLS 1.2 MAY be used. TLS 1.3 (or newer) SHOULD be available on the MASA, but TLS 1.2 MAY be used.
In contrast to [RFC8995] TLS client authentication to the registrar is achieved by using registrar-agent LDevID(RegAgt) credentials instead of pledge IDevID credentials. Consequently BRSKI (pledge-initiator-mode) is distinguishable from BRSKI-PRM (pledge-responder-mode) by the registrar. The registrar SHOULD verify that the registrar-agent is authorized to establish a connection to the registrar by TLS client authentication using LDevID(RegAgt) credentials. If the connection form registrar-agent to registrar is established, the authorization SHALL be verified again based on the agent-signed-data contained in the PVR. This ensures that the pledge has been triggered by an authorized registrar-agent.

The registrar can receive request objects in different formats as defined in [RFC8995]. Specifically, the registrar will receive JSON-in-JWS objects generated by the pledge for voucher-request and enrollment-request (instead of BRSKI voucher-request as CMS-signed JSON and enrollment-request as PKCS#10 objects).

The registrar-agent SHALL send the PVR by HTTP POST to the registrar endpoint: "/.well-known/brski/requestvoucher"

The Content-Type header field for JSON-in-JWS PVR is: application/voucher-jws+json (see Figure 7 for the content definition), as defined in [I-D.ietf-anima-jws-voucher].

The registrar-agent SHOULD set the Accept field in the request-header indicating the acceptable Content-Type for the voucher-response. The voucher-response Content-Type header field SHOULD be set to application/voucher-jws+json as defined in [I-D.ietf-anima-jws-voucher].

After receiving the PVR from registrar-agent, the registrar SHALL perform the verification as defined in section 5.3 of [RFC8995]. In addition, the registrar shall verify the following parameters from the PVR:

* agent-provided-proximity-registrar-cert: MUST contain registrar’s own LDevID(Reg) certificate to ensure the registrar in proximity of the registrar-agent is the destination for this PVR.

* agent-signed-data: The registrar MUST verify that the agent provided data has been signed with the LDevID(RegAgt) credential indicated in the "kid" JOSE header parameter. If the certificate is not included in the agent-sign-cert properties of the PVR, it must be fetched out-of-band by the registrar if "agent-proximity" assertion is requested.
* agent-sign-cert: MAY contain an array of base64-encoded certificate data starting with the LDevID(RegAgt) certificate. If contained the registrar MUST verify that the LDevID(ReAgt) certificate, used to sign the data, is still valid. If the certificate is already expired, the registrar SHALL reject the request. Validity of used signing certificates during bootstrapping is necessary as no trusted timestamp is available, see also Section 9.3.

If the agent-signed-cert is not provided, the registrar MUST fetch the LDevID(RegAgt) certificate, based on the provided SubjectKeyIdentifier (SKID) contained in the kid header of the agent-signed-data, and perform this verification. This requires, that the registrar can fetch the LDevID(RegAgt) certificate data (including intermediate CA certificates if existent) based on the SKID.

If the validation fails the registrar SHOULD respond with HTTP 404 error code to the registrar-agent. HTTP 406 error code SHOULD be used if the format of PVR is unknown.

If the validation succeeds, the registrar SHOULD accept the PVR to join the domain as defined in section 5.3 of [RFC8995]. The registrar then establishes a TLS connection to MASA as described in section 5.4 of [RFC8995] to obtain a voucher for the pledge.

The registrar SHALL construct the payload of the RVR object as defined in [RFC8995]. The RVR object encoding SHALL be JSON-in-JWS as defined in [I-D.ietf-anima-jws-voucher].

The header of the RVR SHALL contain the following parameter as defined in [RFC7515]:

* alg: algorithm used to create the object signature.

* x5c: contains the base64-encoded registrar LDevID certificate(s). It may optionally contain the certificate chain for this certificate.

The payload of the RVR object MUST contain the following parameter as part of the voucher request as defined in [RFC8995]:

* created-on: contains the current date and time in yang:date-and-time format for the RVR creation time.

* nonce: copied form the PVR
* serial-number: contains the pledge product-serial-number. The registrar MUST verify that the IDevID certificate subject X520SerialNumber of the pledge (X520SerialNumber) matches the serial-number value in the PVR. In addition, it MUST be equal to the serial-number value contained in the agent-signed data of PVR.

* assertion: contains the voucher assertion requested by the pledge (agent-proximity). The registrar provides this information to assure successful verification of agent proximity based on the agent-signed-data.

* prior-signed-voucher-request: contains the PVR provided by the registrar-agent.

The RVR can be enhanced optionally with the following parameter as defined in Section 6.1:

* agent-sign-cert: contains the LDevID(RegAgt) certificate or the LDevID(RegAgt) certificate including the certificate chain. In the context of this document it is a JSON array of base64encoded certificate information and handled in the same way as x5c header objects.

If only a single object is contained in the x5c it MUST be the base64-encoded LDevID(RegAgt) certificate. If multiple certificates are included in the x5c, the first MUST be the base64-encoded LDevID(RegAgt) certificate.

The MASA uses this information for verification that the registrar-agent is in proximity to the registrar to state the corresponding assertion "agent-proximity". Note that the agent-sign-cert may also be contained in the "prior-signed-voucher-request" carrying the PVR if the pledge included it.

The object is signed using the registrar LDevID(Reg) credential, which corresponds to the certificate signaled in the JOSE header.
Figure 11: Representation of RVR

The registrar SHALL send the RVR to the MASA endpoint by HTTP POST:
"/.well-known/brski/requestvoucher"

The RVR Content-Type header field is defined in [I-D.ietf-anima-jws-voucher] as: application/voucher-jws+json

The RVR SHOULD set the Accept header indicating the desired media type for the voucher-response. The media type is application/voucher-jws+json as defined in [I-D.ietf-anima-jws-voucher].

Once the MASA receives the RVR it SHALL perform the verification as described in section 5.5 in [RFC8995].

In addition, the following processing SHALL be performed for PVR data contained in RVR "prior-signed-voucher-request" field:
agent-provided-proximity-registrar-cert: The MASA MAY verify that this field contains the LDevID(Reg) certificate. If so, it MUST correspond to the LDevID(Reg) certificate used to sign the RVR. Note: Correspond here relates to the case that a single LDevID(Reg) certificate is used or that different LDevID(Reg) certificates are used, which are issued by the same CA.

agent-signed-data: The MASA MAY verify this field to issue "agent-proximity" assertion. If so, the agent-signed-data MUST contain the pledge product-serial-number, contained in the "serial-number" field of the PVR (from "prior-signed-voucher-request" field) and also in "serial-number" field of the RVR. The LDevID(RegAgt) certificate used to generate the signature is identified by the "kid" parameter of the JOSE header (agent-signed-data). If the assertion "agent-proximity" is requested, the RVR MUST contain the corresponding LDevID(RegAgt) certificate data in the "agent-sign-cert" field of either the LDevID(RegAgt) certificate of RVR or of PVR from "prior-signed-voucher-request" field. It MUST be verified by the MASA that it can be verified to the same domain CA as the LDevID(Reg) certificate.

If the "agent-sign-cert" field is not provided, the MASA MAY state a lower level assertion value, e.g.: "logged" or "verified" Note: Sub-CA certificate(s) MUST also be carried by "agent-sign-cert", in case the LDevID(RegAgt) certificate is issued by a sub-CA and not the domain CA known to the MASA. As the "agent-sign-cert" field is defined as array (x5c), it can handle multiple certificates.

If validation fails, the MASA SHOULD respond with an HTTP error code to the registrar. The HTTP error codes are kept the same as defined in section 5.6 of [RFC8995], and comprise the codes: 403, 404, 406, and 415.

The expected voucher-response format for the pledge-responder-mode the application/voucher-jws+json as defined in [I-D.ietf-anima-jws-voucher] is applied. The voucher syntax is described in detail by [RFC8366]. Figure 12 shows an example of the contents of a voucher.
The MASA returns the voucher-response object to the registrar.

After receiving the voucher the registrar SHOULD evaluate it for transparency and logging purposes as outlined in section 5.6 of [RFC8995]. The registrar MUST add an additional signature to the voucher-response object, by signing it using its registrar credentials (LDevID(Reg)). This signature is done over the same content as the MASA signature of the voucher and provides a proof of possession of the private key corresponding to the LDevID(Reg) the pledge received in the trigger for the PVR (see Figure 5). The registrar MUST use the same LDevID(Reg) credential that is used for authentication in the TLS handshake to authenticate towards the registrar-agent. This ensures that the same LDevID(Reg) certificate can be used to verify the signature as transmitted in the voucher request as is transferred in the PVR in the agent-provided-proximity-registrar-cert component. Figure Figure 13 below provides an example of the voucher with two signatures.
Figure 13: Representation of MASA issued voucher with additional registrar signature

Depending on the security policy of the operator, this signature can also be interpreted by the pledge explicit authorization of the registrar to install the contained trust anchor. The registrar sends the voucher to the registrar-agent.

After receiving the voucher, the registrar-agent sends the PER to the registrar. Deviating from BRSKI the PER is not a raw PKCS#10 object. As the registrar-agent is involved in the exchange, the PKCS#10 is wrapped in a JWS object by the pledge and signed with pledge’s IDevID to ensure proof-of-identity as outlined in Figure 9.
[RFC7030] EST standard endpoints (/simpleenroll, /simplereenroll, /serverkeygen, /cacerts) on the registrar cannot be used for BRSKI-PRM. This is caused by the utilization of signature wrapped-objects in BRSKI-PRM. As EST requires to sent a raw PKCS#10 request to the /simpleenroll endpoint, this document makes an enhancement by utilizing EST but with the exception to transport a signature wrapped PKCS#10 request. Therefore a new endpoint for BRSKI-PRM on the registrar is defined as "/.well-known/brski/requestenroll"

The Content-Type header of PER is: application/jose+json.

This is a deviation from the Content-Type header values used in [RFC7030] and results in additional processing at the domain registrar (as EST server). Note, the registrar is already aware that the bootstrapping is performed in a pledge-responder-mode due to the use of the LDevID(RegAgt) certificate in the TLS establishment and the provided PVR as JSON-in-JWS object.

* If the registrar receives a PER with Content-Type header: application/jose+json, it MUST verify the wrapping signature using the certificate indicated in the JOSE header.

* The registrar verifies that the pledge’s certificate (here IDevID), carried in "x5c" header field, is accepted to join the domain after successful validation of the PVR.

* If both succeed, the registrar utilizes the PKCS#10 request contained in the JWS object body as "P10" parameter of "ietf-sztp-csr:csr" for further processing of the enrollment request with the corresponding domain CA. It creates a registrar-enrollment-request (RER) by utilizing the protocol expected by the domain CA. The domain registrar may either directly forward the provided PKCS#10 request to the CA or provide additional information about attributes to be included by the CA into the requested LDevID certificate. The approach of sending this information to the CA depends on the utilized certificate management protocol between the RA and the CA and is out of scope for this document.

The registrar-agent SHALL send the PER to the registrar by HTTP POST to the endpoint: "/.well-known/brski/requestenroll"

The registrar SHOULD respond with an HTTP 200 in the success case or fail with HTTP 4xx/5xx codes as defined by the HTTP standard.

A successful interaction with the domain CA will result in a pledge LDevID certificate, which is then forwarded by the registrar to the registrar-agent using the Content-Type header: application/pkcs7-mime.
As the pledge will verify its own certificate LDevID certificate when received, it also needs the corresponding CA certificates. This is done in EST using the /cacerts endpoint, which provides the CA certificates over a TLS protected connection. BRSKI-PRM requires a signature wrapped CA certificate response, to avoid that the pledge can be provided with arbitrary CA certificates in an authorized way. The additional signature of the registrar will allow the pledge to verify the authorization to install CA certificates. As the CA certificates are provided to the pledge after the voucher, the pledge has the necessary information to validate the provisioning object.

To allow the registrar-agent to request a signature wrapped CA certificate object, a new endpoint for BRSKI-PRM on the registrar is defined as "/.well-known/brski/wrappedcacerts"

The registrar-agent SHALL requests the EST CA trust anchor database information (in form of CA certificates) with an HTTPS GET message.

The Content-Type header of the response SHALL be: application/jose+json.

This is a deviation from the Content-Type header values used in [RFC7030] and results in additional processing at the domain registrar (as EST server). The additional processing is the signature of the CA certificate information using the LDevID(Reg) credential resulting in a signed JSON object. The CA certificates are provided as base64 encoded x5b.

```json
{
    "payload": {
        "x5b": [ "base64encodedvalue==" ],
        "signatures": [
            {
                "protected": {
                    "alg": "ES256",
                    "x5c": [ "base64encodedvalue==" ]
                },
                "signature": "base64encodedvalue=="
            }
        ]
    }
}
```

Figure 14: Representation of CA certificates data with additional registrar signature
The registrar-agent has now finished the exchanges with the domain registrar and can supply the voucher-response (from MASA via Registrar), the CA certificates, and the enrollment-response (LDevID certificate, from CA via Registrar) to the pledge. It can close the TLS connection to the domain registrar and provide the objects to the pledge(s). The content of the response objects is defined by the voucher [RFC8366] and the certificate [RFC5280].

5.5.3. Response Object Supply by Registrar-Agent to Pledge

The following description assumes that the registrar-agent has obtained the response objects from the domain registrar. It will re-start the interaction with the pledge. To contact the pledge, it may either discover the pledge as described in Section 5.4.2 or use stored information from the first contact with the pledge.

Preconditions in addition to Section 5.5.2:

* Registrar-agent: possesses voucher and LDevID certificate and optionally CA certificates.

![Diagram of response and status handling between pledge and registrar-agent]

The registrar-agent provides the information via distinct pledge endpoints as following.
The registrar-agent SHALL send the voucher-response to the pledge by HTTP POST to the endpoint: "/.well-known/brski/pledge-voucher".

The registrar-agent voucher-response Content-Type header is application/voucher-jws+json and contains the voucher as provided by the MASA. An example if given in Figure 12 for a MASA only signed voucher and in Figure Figure 13 for multiple signatures.

If a single signature is contained, the pledge receives the voucher and verifies it as described in section 5.6.1 in [RFC8995].

A nonceless voucher may be accepted as in [RFC8995] and may be allowed by a manufactures pledge implementation. It requires to perform the validation that the pledge is connected to an authorized registrar-agent by other means, as the registrar would be able to verify it using the agent-signed-data in the PER.

If multiple signatures are contained in the voucher, the pledge SHALL perform the signature verification in the following order:

1. Validate MASA signature as described in section 5.6.1 in [RFC8995] successfully.
2. Install contained trust anchor provisionally.
3. Verify registrar signature as described in section 5.6.1 in [RFC8995] successfully, but take the registrar certificate instead of the MASA certificate for verification.
4. Validate the registrar certificate received in the agent-provided-proximity-registrar-cert in the pledge-voucher-request trigger request (in the field "agent-provided-proximity-registrar-cert") successfully, including validity and authorization to bootstrap the particular pledge.

If all verification steps stated above have been performed successfully, the pledge SHALL end the provisional accept state for the domain trust anchor and the LDevID(Reg). If multiple signatures are contained in the voucher-response, the pledge MUST verify all successfully.

If an error occurs during the verification it SHALL be signaled in the reason field of the pledge voucher status object.
After verification the pledge MUST reply with a status telemetry message as defined in section 5.7 of [RFC8995]. The pledge generates the voucher status object and provides it as JOSE object with the wrapping signature in the response message to the registrar-agent.

The response has the Content-Type application/jose+json and is signed using the IDevID of the pledge as shown in Figure 16. As the reason field is optional (see [RFC8995]), it MAY be omitted in case of success.

```
{
   "payload": {
      "version": 1,
      "status": true,
      "reason": "Informative human readable message",
      "reason-context": {
         "additional": "JSON"
      }
   },
   "signatures": [
      {
         "protected": {
            "alg": "ES256",
            "x5c": [ "base64encodedvalue==" ]
         },
         "signature": "base64encodedvalue=="
      }
   ]
}
```

Figure 16: Representation of pledge voucher status telemetry

The registrar-agent SHALL provide the set of CA certificates requested from the registrar to the pledge by HTTP POST to the endpoint: "/.well-known/brski/pledge-CAcerts".

As the CA certificate provisioning is crucial from a security perspective, this provisioning SHALL only be done, if the voucher-response has been provided to the pledge.

The supply CA certificates message has the Content-Type application/jose+json and is signed using the LDevID(Reg) of the registrar pledge as shown in Figure 14.

The CA certificates are provided as base64 encoded x5b. The pledge SHALL install the received CA certificates in its trust anchor database after successful verification of the registrar’s signature.
If validation of the wrapping signature fails, the pledge SHOULD respond with the HTTP 403 error code. The HTTP 406 error code SHOULD be used, if the response is in an unknown format.

The registrar-agent SHALL send the enroll-response to the pledge by HTTP POST to the endpoint: "/.well-known/brski/pledge-enrollment".

The registrar-agent enroll-response Content-Type header, when using EST [RFC7030] as enrollment protocol between the registrar-agent and the infrastructure, is application/pkcs7-mime. Note that it only contains the LDevID certificate for the pledge, not the certificate chain.

Upon reception, the pledge SHALL verify the received LDevID certificate. The pledge SHALL generate the enroll status object and provide it in the response message to the registrar-agent. If the verification of the LDevID certificate succeeds, the status SHALL be set to true, otherwise to FALSE.

The pledge MUST reply with a status telemetry message as defined in section 5.9.4 of [RFC8995]. As for the other objects, the enrollment status object is provided with an additional signature using JOSE. If the pledge verified the received LDevID certificate successfully it SHALL sign the response using the LDevID of the pledge as shown in Figure 17. In the failure case, the pledge SHALL use the available IdevID credentials. As the reason field is optional, it MAY be omitted in case of success.

The response has the Content-Type application/jose+json.
{  
    "payload": {  
        "version": 1,  
        "status": true,  
        "reason": "Informative human readable message",  
        "reason-context": {  
            "additional": "JSON"  
        }  
    },  
    "signatures": [  
        {  
            "protected": {  
                "alg": "ES256",  
                "x5c": [ "base64encodedvalue==" ]  
            },  
            "signature": "base64encodedvalue=="  
        }  
    ]  
}

Figure 17: Representation of pledge enroll status telemetry

Once the registrar-agent has collected the information, it can connect to the registrar-agent to provide the status responses to the registrar.

5.5.4. Telemetry status handling (registrar-agent - domain registrar)

The following description requires that the registrar-agent has collected the status objects from the pledge. It SHALL provide the status objects to the registrar for further processing.

Preconditions in addition to Section 5.5.2:

* Registrar-agent: possesses voucher status and enroll status objects from pledge.
The registrar-agent MUST provide the collected pledge voucher status object to the registrar. This status indicates if the pledge could process the voucher successfully or not.

If the TLS connection to the registrar was closed, the registrar-agent establishes a TLS connection with the registrar as stated in Section 5.5.2.

The registrar-agent sends the pledge voucher status object without modification to the registrar with an HTTP-over-TLS POST using the registrar endpoint "/.well-known/brski/voucher_status". The Content-Type header is kept as application/jose+json as described in Figure 15 and depicted in the example in Figure 16.

The registrar SHALL verify the signature of the pledge voucher status object and validate that it belongs to an accepted device in his domain based on the contained "serial-number" in the IDevID certificate referenced in the header of the voucher status object.

According to [RFC8995] section 5.7, the registrar SHOULD respond with an HTTP 200 in the success case or fail with HTTP 4xx/5xx codes as defined by the HTTP standard. The registrar-agent may use the response to signal success / failure to the service technician operating the registrar agent. Within the server logs the server SHOULD capture this telemetry information.
The registrar SHOULD proceed with collecting and logging status information by requesting the MASA audit-log from the MASA service as described in section 5.8 of [RFC8995].

The registrar-agent MUST provide the pledge’s enroll status object to the registrar. The status indicates the pledge could process the enroll-response object and holds the corresponding private key.

The registrar-agent sends the pledge enroll status object without modification to the registrar with an HTTP-over-TLS POST using the registrar endpoint "/.well-known/brski/enrollstatus". The Content-Type header is kept as application/jose+json as described in Figure 15 and depicted in the example in Figure 17.

The registrar MUST verify the signature of the pledge enroll status object. Also, the registrar SHALL validate that the pledge belongs to an accepted device in his domain based on the contained product-serial-number in the LDevID certificate referenced in the header of the enroll status object. The registrar SHOULD log this event. In case the enroll status object indicates a failure, the pledge was unable to verify the received LDevID certificate and therefore signed the enroll status objects with its IDevID credential. Note that the verification of a signature of the object is a deviation from the described handling in section 5.9.4 of [RFC8995].

According to [RFC8995] section 5.9.4, the registrar SHOULD respond with an HTTP 200 in the success case or fail with HTTP 4xx/5xx codes as defined by the HTTP standard. Based on the failure case the registrar MAY decide that for security reasons the pledge is not allowed to reside in the domain. In this case the registrar MUST revoke the certificate. The registrar-agent may use the response to signal success / failure to the service technician operating the registrar agent. Within the server log the registrar SHOULD capture this telemetry information.

6. Artifacts

6.1. Voucher Request Artifact

The following enhancement extends the voucher-request as defined in [RFC8995] to include additional fields necessary for handling bootstrapping in the pledge-responder-mode.
6.1.1. Tree Diagram

The following tree diagram is mostly a duplicate of the contents of
[RFC8995], with the addition of the fields agent-signed-data, the
registrar-proximity-certificate, and agent-signing certificate. The
tree diagram is described in [RFC8340]. Each node in the diagram is
fully described by the YANG module in Section Section 6.1.2.

module: ietf-voucher-request-prm

grouping voucher-request-prm-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
      +-- prior_signed_voucher_request? binary
      +-- proximity_registrar_cert?    binary
      +-- agent_signed_data?           binary
      +-- agent_provided_proximity_registrar_cert? binary
      +-- agent_sign_cert?             binary

6.1.2. YANG Module

The following YANG module extends the [RFC8995] Voucher Request to
include a signed artifact from the registrar-agent (agent-signed-
data) as well as the registrar-proximity-certificate and the agent-
signing certificate.

<CODE BEGINS> file "ietf-voucher-request-prm@2022-07-05.yang"
module ietf-voucher-request-prm {
  yang-version 1.1;

  prefix vrprm;

  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-request {
  prefix vcr;
  description
    "This module defines the format for a voucher request, which is produced by a pledge as part of the RFC8995 onboarding process."
  reference
    "RFC 8995: Bootstrapping Remote Secure Key Infrastructure";
}

organization
  "IETF ANIMA Working Group";

contact
  "WG Web:  <http://tools.ietf.org/wg/anima/>
  WG List:  <mailto:anima@ietf.org>
  Author:   Steffen Fries
            <mailto:steffen.fries@siemens.com>
  Author:   Eliot Lear
            <mailto: lear@cisco.com>
  Author:   Thomas Werner
            <mailto: thomas-werner@siemens.com>
  Author:   Michael Richardson
            <mailto: mcr+ietf@sandelman.ca>"

description
  "This module defines the format for a voucher-request form the pledge in responder mode. It bases on the voucher-request defined in RFC 8995, which is a superset of the voucher itself. It provides content to the MASA for consideration during a voucher-request.

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 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

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

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info).
This version of this YANG module is part of RFC xxxx; see the RFC itself for full legal notices.

revision 2022-07-05 {
  description
    "Initial version";
  reference
    "RFC XXXX: BRSKI for Pledge in Responder Mode";
}

// Top-level statement
rc:yang-data voucher-request-prm-artifact {
  // YANG data template for a voucher-request.
  uses voucher-request-prm-grouping;
}

// Grouping defined for future usage
grouping voucher-request-prm-grouping {
  description
    "Grouping to allow reuse/extension in future work.";
  uses vcr:voucher-request-grouping {
    augment voucher {
      description "Base the voucher-request-prm upon the regular one";
      leaf agent-signed-data {
        type binary;
        description
          "The agent-signed-data field contains a JOSE [RFC7515] object provided by the Registrar-Agent to the Pledge.

          This artifact is signed by the Registrar-Agent and contains a copy of the pledge’s serial-number.";
      }
    }

    leaf agent-provided-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 X.690.

        The first certificate in the registrar TLS server certificate_list sequence (the end-entity TLS certificate; see RFC 8446) presented by the registrar to the registrar-agent and provided to the pledge.

        This MUST be populated in a pledge’s voucher-request...";
    }
  }
}
when an agent-proximity assertion is requested.

reference

"ITU X.690: Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

leaf-list agent-sign-cert {
  type binary;
  min-elements 1;
  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 X.690.
  This certificate can be used by the pledge, the registrar, and the MASA to verify the signature of agent-signed-data. It is an optional component for the pledge-voucher request.
  This MUST be populated in a registrar’s voucher-request when an agent-proximity assertion is requested.
  It is defined as list to enable inclusion of further certificates along the certificate chain if different issuing CAs have been used for the registrar-agent and the registrar."

  reference
  "ITU X.690: Information Technology - ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)
RFC 5280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile";

}

Examples for the PVR are provided in Section 5.5.2.
7. IANA Considerations

This document requires the following IANA actions.

7.1. BRSKI well-known Registry

IANA is requested to enhance the Registry entitled: "BRSKI Well-Known URIs" with the following endpoints:

<table>
<thead>
<tr>
<th>URI</th>
<th>Description</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>pledge-voucher-request</td>
<td>create pledge-voucher-request</td>
<td>[THISRFC]</td>
</tr>
<tr>
<td>pledge-enrollment-request</td>
<td>create pledge-enrollment-request</td>
<td>[THISRFC]</td>
</tr>
<tr>
<td>pledge-voucher</td>
<td>supply voucher response</td>
<td>[THISRFC]</td>
</tr>
<tr>
<td>pledge-enrollment</td>
<td>supply enrollment response</td>
<td>[THISRFC]</td>
</tr>
<tr>
<td>pledge-cacerts</td>
<td>supply CA certificates to pledge</td>
<td>[THISRFC]</td>
</tr>
<tr>
<td>requestenroll</td>
<td>supply PER to registrar</td>
<td>[THISRFC]</td>
</tr>
<tr>
<td>wrappedcacerts</td>
<td>request wrapped CA certificates</td>
<td>[THISRFC]</td>
</tr>
</tbody>
</table>

8. Privacy Considerations

The credential used by the registrar-agent to sign the data for the pledge in case of the pledge-initiator-mode should not contain personal information. Therefore, it is recommended to use an LDevID certificate associated with the device instead of a potential service technician operating the device, to avoid revealing this information to the MASA.

9. Security Considerations

9.1. Exhaustion Attack on Pledge

Exhaustion attack on pledge based on DoS attack (connection establishment, etc.)

9.2. Misuse of acquired Voucher and Enrollment objects by Registrar-Agent

A Registrar-agent that uses acquired voucher and enrollment objects for domain-A in domain-B can be avoided by the PVR processing on the domain registrar side. This requires the domain registrar to verify the "proximity-registrar-cert" field in the PVR against his own LDevID(Reg). In addition, the domain registrar has to verify the association of the pledge to his domain based on the product-serial-number contained in the PVR and in the pledge IDevID certificate. Moreover, the registrar verifies if the registrar-agent is authorized to interact with the pledge for voucher-requests, based on the LDevID(RegAgt) certificate information contained in the PVR.
Misbinding of a pledge by a faked domain registrar is countered as described in BRSKI security considerations (section 11.4).

9.3. Misuse of Registrar-Agent Credentials

Concerns on opportunities to misuse the registrar-agent with a valid LDevID, may be addressed by utilizing short-lived certificates (e.g., valid for a day) to authenticate the registrar-agent against the domain registrar. The LDevID(RegAgt) certificate may be acquired by a prior BRSKI run for the registrar-agent, if IDevID is available on registrar-agent. Alternatively, the LDevID may be acquired by a service technician from the domain PKI system.

In addition it is required that the LDevID(RegAgt) certificate is valid for the complete bootstrapping phase. This avoids a registrar-agent could be misused to create arbitrary "agent-signed-data" objects to perform an authorized bootstrapping of a rouge pledge. As "agent-signed-data" could be dated after the validity time of the LDevID(RegAgt) certificate, due to missing trusted timestamp in the registrar-agents signature. To address this, the registrar SHOULD verify the certificate used to create the signature on "agent-signed-data". Furthermore the registrar also verifies the LDevID(RegAgt) certificate used in the TLS handshake. If both certificates are successfully verified, the registrar-agents signature can be considered as valid.

9.4. YANG Module Security Considerations

The enhanced voucher-request described in section Section 6.1 is bases on [RFC8995], but uses a different encoding based on [I-D.ietf-anima-jws-voucher]. Therefore similar considerations as described in Section 11.7 (Security Considerations) of [RFC8995] apply. The YANG module specified in this document defines the schema for data that is subsequently encapsulated by a JOSE signed-data Content-type as described in [I-D.ietf-anima-jws-voucher]. As such, all of the YANG-modeled data is protected against modification. The use of YANG to define data structures via the "yang-data" statement, is relatively new and distinct from the traditional use of YANG to define an API accessed by network management protocols such as NETCONF [RFC6241] and RESTCONF [RFC8040]. For this reason these guidelines do not follow the template described by Section 3.7 of [RFC8407].

10. Acknowledgments

We would like to thank the various reviewers, in particular Brian E. Carpenter, Oskar Camenzind, and Hendrik Brockhaus for their input and discussion on use cases and call flows.
11. References

11.1. Normative References

[I-D.ietf-anima-jws-voucher]

[I-D.ietf-anima-rfc8366bis]

[I-D.ietf-netconf-sztp-csr]


11.2. Informative References


Appendix A.  History of Changes [RFC Editor: please delete]

From IETF draft 03 -> IETF draft 04:

* Simplified YANG definition by augmenting the voucher request from RFC 8995 instead of redefining it.

* Added explanation for terminology "endpoint" used in this document, issue #16
* Added clarification that registrar-agent may collect PVR or PER or both in one run, issue #17

* Added a statement that nonceless voucher may be accepted, issue #18

* Simplified structure in section Section 3.1, issue #19

* Removed join proxy in Figure 1 and added explanatory text, issue #20

* Added description of pledge-CAcerts endpoint plus further handling of providing a wrapped CA certs response to the pledge in section Section 5.5.3; also added new required registrar endpoint (section Section 5.5.2 and IANA considerations) for the registrar to provide a wrapped CA certs response, issue #21

* utilized defined abbreviations in the document consistently, issue #22

* Reworked text on discovery according to issue #23 to clarify scope and handling

* Added several clarifications based on review comments

From IETF draft 02 -> IETF draft 03:

* Updated examples to state "base64encodedvalue==" for x5c occurrences

* Include link to SVG graphic as general overview

* Restructuring of section 5 to flatten hierarchy

* Enhanced requirements and motivation in Section 4

* Several editorial improvements based on review comments

From IETF draft 01 -> IETF draft 02:

* Issue #15 included additional signature on voucher from registrar in section Section 5.5.2 and section Section 5.2 The verification of multiple signatures is described in section Section 5.5.3

* Included representation for General JWS JSON Serialization for examples

* Included error responses from pledge if it is not able to create a pledge-voucher request or an enrollment request in section Section 5.5.1

* Removed open issue regarding handling of multiple CSRs and enrollment responses during the bootstrapping as the initial target it the provisioning of a generic LDevID certificate. The defined endpoint on the pledge may also be used for management of further certificates.

From IETF draft 00 -> IETF draft 01:

* Issue #15 lead to the inclusion of an option for an additional signature of the registrar on the voucher received from the MASA before forwarding to the registrar-agent to support verification of POP of the registrars private key in section Section 5.5.2 and Section 5.5.3.

* Based on issue #11, a new endpoint was defined for the registrar to enable delivery of the wrapped enrollment request from the pledge (in contrast to plain PKCS#10 in simple enroll).

* Decision on issue #8 to not provide an additional signature on the enrollment-response object by the registrar. As the enrollment response will only contain the generic LDevID certificate. This credential builds the base for further configuration outside the initial enrollment.

* Decision on issue #7 to not support multiple CSRs during the bootstrapping, as based on the generic LDevID certificate the pledge may enroll for further certificates.

* Closed open issue #5 regarding verification of ietf-ztp-types usage as verified via a proof-of-concept in section "#exchanges_uc2_1".

* Housekeeping: Removed already addressed open issues stated in the draft directly.

* Reworked text in from introduction to section pledge-responder-mode

* Fixed "serial-number" encoding in PVR/RVR

* Added prior-signed-voucher-request in the parameter description of the registrar-voucher-request in Section 5.5.2.
**Note added in Section 5.5.2 if sub-CAs are used, that the corresponding information is to be provided to the MASA.**

**Inclusion of limitation section (pledge sleeps and needs to be waked up. Pledge is awake but registrar-agent is not available) (Issue #10).**

**Assertion-type aligned with voucher in RFC8366bis, deleted related open issues. (Issue #4)**

**Included table for endpoints in Section 5.3 for better readability.**

**Included registrar authorization check for registrar-agent during TLS handshake in section Section 5.5.2. Also enhanced figure Figure 10 with the authorization step on TLS level.**

**Enhanced description of registrar authorization check for registrar-agent based on the agent-signed-data in section Section 5.5.2. Also enhanced figure Figure 10 with the authorization step on pledge-voucher-request level.**

**Changed agent-signed-cert to an array to allow for providing further certificate information like the issuing CA cert for the LDevID(RegAgt) certificate in case the registrar and the registrar-agent have different issuing CAs in Figure 10 (issue #12). This also required changes in the YANG module in Section 6.1.2**

**Addressed YANG warning (issue #1)**

**Inclusion of examples for a trigger to create a pledge-voucher-request and an enrollment-request.**

**From IETF draft-ietf-anima-brski-async-enroll-03 -> IETF anima-brski-prm-00:**

**Moved UC2 related parts defining the pledge in responder mode from draft-ietf-anima-brski-async-enroll-03 to this document This required changes and adaptations in several sections to remove the description and references to UC1.**

**Addressed feedback for voucher-request enhancements from YANG doctor early review in Section 6.1 as well as in the security considerations (formerly named ietf-async-voucher-request).**
* Renamed ietf-async-voucher-request to IETF-voucher-request-prm to allow better listing of voucher related extensions; aligned with constraint voucher (#20)

* Utilized ietf-voucher-request-async instead of ietf-voucher-request in voucher exchanges to utilize the enhanced voucher-request.

* Included changes from draft-ietf-netconf-sztp-csr-06 regarding the YANG definition of csr-types into the enrollment request exchange.

From IETF draft 02 -> IETF draft 03:

* Housekeeping, deleted open issue regarding YANG voucher-request in Section 5.5.1 as voucher-request was enhanced with additional leaf.

* Included open issues in YANG model in Section 5.1 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 Section 5.5.

* 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 Section 5.1.

* 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 Section 5.5.

* Details on trust relationship between registrar-agent and pledge (issue #5) included in Section 5.1.

* Split of use case 2 call flow into sub sections in Section 5.5.

From IETF draft 00 -> IETF draft 01:

* Update of scope in Section 3.1 to include in which the pledge acts as a server. This is one main motivation for use case 2.

* Rework of use case 2 in Section 5.1 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 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 new section 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 Section 5.1, e.g. to accommodate distribution of CA certificates.

* Updated CMP example in 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 4. 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 UC1 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-contain 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-PRM call flow.

From individual version 01 -> 02:

* Update of introduction text to clearly relate to the usage of IDevID and LDevID.

* Update of description of architecture elements and changes to BRSKI in Section 5.

* Enhanced consideration of existing enrollment protocols in the context of mapping the requirements to existing solutions in Section 4.

From individual version 00 -> 01:

* Update of examples, specifically for building automation as well as two new application use cases in Section 3.1.
* 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 5.

* Enhancement of description of architecture elements and changes to BRuki in Section 5.

* Consideration of existing enrollment protocols in the context of mapping the requirements to existing solutions in Section 4.

* New section starting with the mapping to existing enrollment protocols by collecting boundary conditions.

Authors’ Addresses

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

Thomas Werner
Siemens AG
Otto-Hahn-Ring 6
81739 Munich
Germany
Email: thomas-werner@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

Michael C. Richardson
Sandelman Software Works
Email: mcr+ietf@sandelman.ca
URI: http://www.sandelman.ca/
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 12 January 2023.

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. Voucher Artifact with JSON Web Signature .......... 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) ................... 13
Authors’ Addresses .................................... 15

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 the serialization to JSON [RFC8259] with a CMS signature according to [RFC5652]. The resulting Voucher artifact has the media type "application/voucher-cms+json".

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 JSON Web Signature (JWS) [RFC7515]. The encoding specified in this document is used by [I-D.ietf-anima-brski-prm] and may be preferred for use cases requiring signed JSON objects.

This document does not extend the YANG definition of [RFC8366].

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, [BRISKI] provides this context via the MIME-Type for the voucher artifact.

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. Voucher Artifact with JSON Web Signature

The voucher [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 The following serializations are defined:[RFC7515] in section 3.2 and section 7 provides more details.

1. "JWS Compact Serialization", [RFC7515] section 7.1
2. "JWS JSON Serialization" in, [RFC7515] section 7.2
   - "General JWS JSON Serialization Syntax", [RFC7515] section 7.2.1
This document makes use of the "General JWS JSON Serialization Syntax" to support multi signatures.

There are a number of attributes. They are:

3.1. Unprotected Header

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
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"
* Added optional "typ": "voucher-jws+json" header parameter to JWS objects

8. References

8.1. Normative References


8.2. Informative References
[I-D.ietf-anima-brski-prm]

[I-D.ietf-anima-constrained-voucher]

[I-D.kuehlewind-update-tag]


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].

```json
{
  "payload":
  "eyJpZXRmLXZvdWNoXZItcmVxdWVzdDp2b3VjaGVyIjp7InNlcmlhbCIudW1iZXIiOiIwMDU2Nzg5Iiwibiwibm9uY2U1IiI6MC4xOTBaIiwicHJveGltaXR5LXZja1ZjdHJhci11NXJpIjoiUTAiLCJjcmVhdGVkLW9uIjoiMTIyMDAwNi0xOS0xOSQwOi8xNTk3IiwicmVxdWVzdC1jZXN0IjoiMjAxMTM0MzI0IiwiZnVuZGluZ3MiOiJ2b3VjaGVyIiwic2VjdXJyZXoiOiJ2b3VjaGVyIn0",
  "signatures": [
    {
      "protecte":
      "eyJ4NWMiOlsiTUlJQitUQ0NBYUnQxJQkFmSnUdBGV5wanVNUN1Bb0dDQ3FU0000OUJBUTkNRDB4Q3pBSkJnTlzCQV1UQwtGUK1SVxdFdi1EVI1RS0RBeEhVzVuU21sAVovbW40QDpPmBvKntTlzCQUNRGtwcGjtZEtWhZvVdkWemRFTkJNQ0FYFRjEeE1EWWdOREEmXkRzeE5Gb11Eems1tIRreE1qTXhNak0x0tIRVNdqQfLaNUXN3Qf2RFPZRU0dF0pCVRVFVkJCtUdBMVVFQ2d3TNtihnAmHBWymM1kRG1ZsNdNUk13VFZFRZRU0Zfd293TVRejK5EFTjQomc1TVjzdOZWURWdUFEREE1S2FXNW5tbWxIjBsbGRtbGpaeVEJaTUNJR0J5cUdTTQQ5QWdFR0NDcvUttQ5QXdfSEEewUFSCQz5bGlWerner & Richardson Expires 12 January 2023 [Page 9]"```
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":
  "eyJpZXRmLXZvdWNoZXItcmVxdWVzdDp2b3VjaG5yIjpi7InNlcm1hbC1udWl1Zi1iOi1tM1Z2Nzg5L1wiWRImlkLWl1Zi1cV1ci16IkJCZ3dGbO9VFKTTNhLz1MK1NpNk5EQ09Ea1RsKy9CeGhzP1s1m5vbmnLljo
  "signature":
  "abVg4TDGzSTjVHkQlNeIW3ABu5ZXdMl1cEqwcIA1HFW4Br1Gb0-DRTK
  "figCOGxSW49-ktJcrV1YgKqC4xmZoyQ"
}
```

Figure 2: Example Pledge Voucher Request - PVR
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":
  "eyJpZXRmLXZvdWN0ZXI6dm91Y2hlc3I6eyJ4NmVwN1lsbU5lYl91cGxvc2UtY29tIiwicmFna2xkZGF0YSI6ICJuYW1lIn0.

"signatures":
  "eyJ4NmVwN1lsbU5lYl91cGxvc2UtY29tIiwicmFna2xkZGF0YSI6ICJuYW1lIn0.

Figure 4: Example Voucher Response
Authors’ Addresses

Thomas Werner
Siemens AG
Email: thomas-werner@siemens.com

Michael Richardson
Sandelman Software Works
Email: mcr+ietf@sandelman.ca
An Autonomic Mechanism for Resource-based Network Services Auto-deployment
draft-ietf-anima-network-service-auto-deployment-02

Abstract

This document specifies an autonomic mechanism for resource-based network services deployment through the Autonomic Control Plane (ACP) in a network. This mechanism uses the GeneRic Autonomic Signaling Protocol (GRASP) in [RFC8990] to exchange the information among the autonomic nodes so that the resource along 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 9 January 2023.

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.
1. Introduction

With network development, a class of services with resource requirements (such as bandwidth, queue, and priority) are already emerging, such as video, VR, AR, and so on. To ensure the proper operation of these services, the network needs to allocate sufficient resources for the services in advance. An autonomous network SHOULD have an appropriate mechanism to negotiate the network resources.

From the network perspective, this kind of service has a source IP address and a destination IP address. Therefore, once the kind of service is delivered by a domain network, this service clearly has an access node and a departure node in the network. In an autonomous resource negotiation mechanism, the resources are negotiated between the access node and departure node.

The core goal of this document is to establish an automatic negotiation mechanism to achieve the negotiation and distribution of network resources in the domain network between the service client.
and the network. That is, the server client negotiates with the network how many resources can be provided for specific services in the domain network to support the transmission of network services. The benefits of doing so mainly include the following aspects:

* The resource-based network services auto-deployment satisfies the QoS requirements of the service. If the service wants to ensure its own transmission quality, the most effective solution is to reserve enough transmission resources for the service before the service starts.

* The mechanism of supporting multiple rounds of negotiations enables the service client to change the resource requirements according to the state of the network. For example, when the service is applying for resources, if the network is crowded, Video Conference services can reduce the quality of video to ensure the most basic connectivity. In the traditional network resource reservation, the network will just inform the service that the resource is insufficient and the deployment fails.

* The mechanism can ensure that the resources in the network can be used more efficiently, provide different levels of network resources for different levels of services, and give priority to the network resource requirements of high importance services.

The resource information negotiated in this document is more extensive. Not only negotiation bandwidth resources but also includes and is not limited to queue, priority and other resources. On the one hand, in recent years, the requirements of services for the network have become more complex. Services usually require the network to ensure not only the deterministic bandwidth but also the deterministic end-to-end delay and jitter, so as to deliver the data message to the destination "in time" and "on time". For example, in the telemedicine scenario, in order to ensure that doctors do not feel obvious delay and jitter, it is required that the end-to-end delay should not exceed 20ms. On the other hand, with the development of technology, the network has more refined the scheduling of transmission capacity, and also hopes to open its own capacity to the service clients. The negotiation resources established in this document should support not only negotiating the existing supported resources but also to retain some scalability for the negotiation ability in the future.

This document completes the resource-based self-adaptation among service and network nodes via GRASP. This document defines an autonomic technical objective for resource-based network services auto-deployment. It shows how the ANI can be applied to negotiate resource information for network service auto-deployment. This
document reduces the difficulty of manual operation, avoids the problems of specification limitation and slow response speed in the centralized system, improves the efficiency of service deployment and makes more rational use of network resources. 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.

2. 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] .

3. Terminology & Abbreviations

This document uses terminology defined in [RFC7575].

RRM ASA: Requester ResourceManager ASA. A kind of ResourceManager ASA which starts to request resource in the network.

PRM ASA: Provider ResourceManager ASA. A kind of ResourceManager ASA which provides resource in the network.

APE: Access Provider Edge is the first access provider edge where the service initiator connects to the network or where the path-dependent and resource-based network service starts.

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

Transmit node: A transmit node in the domain network.

ASBR: AS Border Router 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.
Figure 1 shows the process of the auto deployment mechanism. And 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 needs reserve resources for specific, it can request more resources 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 resources should first discover peers that may be able to provide extra resources. The ASA should send out a GRASP Discovery message that contains a ResourceManager Objective option to discover peers also supporting that option.
A GRASP 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 a ResourceManager ASA, the device ignores this message. Further details of the discovery process are described in Section 2.5.4 of [RFC8990].

4.2. Resource Negotiation

After the discovery step, the RRM ASA (Requesting ResourceManager ASA) will act as a GRASP negotiation initiator by sending a GRASP Request message with a ResourceManager Objective option. The RRM ASA indicates in this option the value of the requested resource. ResourceManager GRASP Objective allows multiple types of resources to be requested simultaneously.

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

As described in Section 2.8.8 of [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 removes the acceptable resource from its resource pool and synchronizes to other RRM ASAs and PRM ASAs. The requesting device may use the negotiated resource after receiving synchronization message 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, ?objective-value]

objective-name = "ResourceManager"

objective-flags = uint .bits objective-flag ; as in the GRASP specification

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

The 'objective-value' field expresses the actual value of a negotiation or synchronization objective. So a new objective-value named n-s-deployment-value is defined for Network Service Auto-deployment as follows. The autonomic node can know that it is serving Network Service Auto-deployment according to the objective-value after receiving the GRASP message. The 'objective value' contains two parts, one represents the information of the service itself, and the other represents the requirements of resources.

objective-value = n-s-deployment-value ; An n-s-deployment-value is defined as Figure-2.
n-s-deployment-value
  + service-information
    + source-ip-address
    + destination-ip-address
    + service-tag
  + resource-information
    + resource-requirement-pair
      + resource-type
      + resource-value

Figure-2: Format of n-s-deployment-value

service-information = [ source-ip-address, destination-ip-address, service-tag ]

The source-ip-address and the destination-ip-address represent the source address and destination address. IPv4 and IPv6 addresses are allowed.

resource-information = [ resource-requirement-pair 1, resource-requirement-pair 2, ..., resource-requirement-pair n ]

Resource requirements of different types can be described in an objective option. The ResourceManager objective option supports multi-faceted resource requirements and negotiation.

resource-requirement-pair = [ resourcetype, resval ]

resourcetype /= 0...16; requested or offered resource type, such as bandwidth, queue, and priority.

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(SI), Service Terminator(ST), RRM ASA, PRM ASA and even ASBR.

The service initiator is the resource demander, which ensures the connection of services through negotiation resources with ResourceManager ASA in the domain network. Service Terminator is the end of service. APE represents the first access provider edge where the service initiator connects to the network or where the path-dependent and resource-based network service starts. There may be
multiple Transmit nodes between APE and Service Terminator in the network or even cross multiple network domains through ASBRs. RRM ASA starts a negotiation process to get enough resources in the network. After RRM ASA gets the result about the resource, it sends a response message to the Service Initiator. And PRM ASA manages resources from APE to ST hop-by-hop.

6.1. An example of End-to-End Service

In an End-to-End service, Service Initiator is a kind of access terminal of the network. And the End-to-End service initiator uses ResourceManager ASA to negotiate resources with the ResourceManager ASA in the APE. Figure 3 shows the architecture of the End-to-End service. In the figure, the RRM ASA in SI will act as a GRASP negotiation initiator by sending a GRASP Request message with a ResourceManager Objective option. The RRM ASA indicates in this option the value of the requested resource. When this RRM 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 with the resource value offered to the PRM ASA.

```
+---------+ Negotiation Resource +---------+
| RRM ASA |<------------------------->| PRM ASA |
+---------+                        +---------+    +------+    +------+
|   SI    | ------------------------> |   APE   |--->| Node |--->|  ST  |
+---------+   Transmit data     +---------+    +------+    +------+
```

Figure-3: An example of End-to-End Service

PRM ASA processes receive resource requests and ensure the nodes resource it can manage. When the RRM ASA receives a Negotiation response message, it should check whether the resource value in the Negotiate message is the same as the resource value requested. If it is the same, the RRM ASA should send GRASP Negotiation End messages indicating that the negotiation was successful. If it is not the same, the RRM ASA should communicate with the Service Initiator about the result and decide whether to accept this negotiation. If accepting this negotiation, RRM ASA 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.
Figure 4 shows an example of negotiation process. In the first negotiation round, RRM ASA want to get 10 Mbit/s bandwidth from IP_A to IP_B, so the resource value is [0,10]. Zero means the resource type is bandwidth and ten is the value. The message also include source and target IP and some service information. PRM ASA can use service_tag to decide whether to provide these resources to services in the case of resource constraints. When PRM ASA receives the message, if the PRM ASA think the network can offer this message to RRM ASA, it will response the ACCEPT. But if it not response, it will give the RRM ASA request about how much resource it can offer. In this example, PRM ASA can offer 8 Mbit/s to RRM ASA. The next step, RRM ASA needs to decide whether to change its resource requirements according to the reply. And sends a next round negotiation. And then PRM ASA deal the new resource requirement, it then the requirement it can offer. So it will response ACCEPT. This is an example about how ASAs negotiation resource. And after negotiation, PRM ASA will build a hop-by-hop resource path for service. And Service send data packet through this path.

Figure-4: Example of Negotiation Process

6.2. An example of multiple rounds

In the process of automatic resource management mechanism, RRM ASA and PRM ASA are allowed to negotiate resources for multiple rounds. A very common situation is that the network resources can not meet the resources required by the service, but the service is willing to reduce its resource requirements to ensure the successful deployment of the service. The PRM ASA using Resource Management Objectives contains the resources that the network can provide to the service present in the response message. The RRM ASA changes the resource requirements according to the specific requirements of the received resources and services, to carry out the next round of service negotiation.
6.3. An example of multiple domain network

In a multiple network, PRM ASA doesn’t have the resource status of other domains. So PRM ASA should negotiate with ASBR PRM ASA before response RRM ASA. The PRM ASA should send a Confirm Waiting message to the RRM ASA, to extend its timeout. When the new resource becomes available confirmed by ASBR, the PRM ASA responds with a GRASP Negotiate message with a resource value offered. The process shows in Figure 5. The Confirm Waiting message is described in Section 2.8.9 of [RFC8990].

+----------+           +---------+
| RRM ASA  |<--------->| PRM ASA |
+----------+           +---------+        +--------------+
        |                 | RRM ASA |<------>| ASBR PRM ASA |
        +---------+        +--------------+
Adjust Negotiation Message
Request Negotiation Message
Waiting message
Request Negotiation Message
Negotiation message
Negotiation message

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

Figure-5: An example of a path-based resource negotiation

6.4. An example of changing resource requirements

After the process of automatic resource management mechanism, RRM ASA and PRM ASA are allowed to change and negotiate the resource requirements. In the course of using network services, there will be service requirement change which will lead to the problem of network resource requirement change. ResourceManager ASA needs to be able to handle resource changes in a timely manner to meet service requirements.

During the renegotiation process, RRM ASA resends the service’s resource requirements by using ResourceManager GRASP Objective. And the resource renegotiation process does not require the use of the same PRM ASA as at the last negotiation on the mission. PRM ASA receives the resource negotiation message and makes the determination. If the resource requirements are lower than those
allocated, the response confirms the information and releases the excess resources. If more resources are required than have been allocated, the resource negotiation process follows Section 6.1.

PRM ASA does not change existing resource allocation until negotiation on resource changes is complete. After negotiation, PRM ASA makes changes to the resource pool by using response to the negotiated resource requirements and synchronizes them with other ASA nodes.

6.5. An example of releasing resource requirements

After the service is completed, a mechanism is needed to release network resources so that network resources can be used more efficiently. This process can be seen as a change in resource requirements negotiation, where the resource requirements of the service to the network become zero. A negotiation with PRM ASA was initiated by RRM ASA in SI to reduce the resource footprint of the service. Upon completion of the negotiation, PRM ASA released the resources occupied by the service.

7. Compatibility with Other Technologies

This auto deployment mechanism support multi-round negotiation mechanism which improves the deployment success rate. This mechanism is not available in other protocols.

A gateway device is used between the GRASP network and the MPLS network. As is known, the RSVP belongs to the distribution 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",

[I-D.ietf-spring-segment-routing] "Segment Routing Architecture",

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119,
DOI 10.17487/RFC2119, March 1997,

[RFC8990] "GeneRic Autonomic Signaling Protocol (GRASP)",

[RFC8993] "A Reference Model for Autonomic Networking",

[RFC8994] "GeneRic Autonomic Signaling Protocol (GRASP)",

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
A Voucher Artifact for Bootstrapping Protocols
draft-ietf-anima-rfc8366bis-00

Abstract

This document defines a strategy to securely assign a pledge to an owner using an artifact signed, directly or indirectly, by the pledge’s manufacturer. This artifact is known as a "voucher".

This document defines an artifact format as a YANG-defined JSON document that has been signed using a Cryptographic Message Syntax (CMS) structure. Other YANG-derived formats are possible. The voucher artifact is normally generated by the pledge’s manufacturer (i.e., the Manufacturer Authorized Signing Authority (MASA)).

This document only defines the voucher artifact, leaving it to other documents to describe specialized protocols for accessing it.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the Autonomic Networking Integrated Model and Approach Working Group mailing list (anima@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/anima/.

Source for this draft and an issue tracker can be found at https://github.com/anima-wg/voucher.

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 July 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 ........................................... 3
3. Requirements Language ................................. 5
4. Survey of Voucher Types ............................... 5
5. Voucher Artifact ....................................... 7
   5.1. Tree Diagram .................................... 7
   5.2. Examples ........................................ 8
   5.3. YANG Module ...................................... 9
   5.3.1. "iana-voucher-assertion-type" Module .......... 9
   5.3.2. "ietf-voucher" Module ......................... 11
   5.4. CMS Format Voucher Artifact .................... 15
5. Design Considerations .................................. 16
   6.1. Renewals Instead of Revocations ................. 16
   6.2. Voucher Per Pledge ................................ 17
7. Security Considerations ............................... 17
   7.1. Clock Sensitivity ............................... 18
   7.2. Protect Voucher PKI in HSM .................... 18
   7.3. Test Domain Certificate Validity When Signing .... 18
   7.4. YANG Module Security Considerations ........... 18
8. IANA Considerations ................................... 19
1. Introduction

This document defines a strategy to securely assign a candidate device (pledge) to an owner using an artifact signed, directly or indirectly, by the pledge's manufacturer, i.e., the Manufacturer Authorized Signing Authority (MASA). This artifact is known as the "voucher".

The voucher artifact is a JSON [RFC8259] document that conforms with a data model described by YANG [RFC7950], is encoded using the rules defined in [RFC8259], and is signed using (by default) a CMS structure [RFC5652].

The primary purpose of a voucher is to securely convey a certificate, the "pinned-domain-cert", that a pledge can use to authenticate subsequent interactions. A voucher may be useful in several contexts, but the driving motivation herein is to support secure bootstrapping mechanisms. Assigning ownership is important to bootstrapping mechanisms so that the pledge can authenticate the network that is trying to take control of it.

The lifetimes of vouchers may vary. In some bootstrapping protocols, the vouchers may include a nonce restricting them to a single use, whereas the vouchers in other bootstrapping protocols may have an indicated lifetime. In order to support long lifetimes, this document recommends using short lifetimes with programmatic renewal, see Section 6.1.

This document only defines the voucher artifact, leaving it to other documents to describe specialized protocols for accessing it. Some bootstrapping protocols using the voucher artifact defined in this document include: [ZERO-TOUCH], [SECUREJOIN], and [BRSKI]).

2. Terminology

This document uses the following terms:

Artifact: Used throughout to represent the voucher as instantiated
in the form of a signed structure.

Domain: The set of entities or infrastructure under common administrative control. The goal of the bootstrapping protocol is to enable a pledge to discover and join a domain.

Imprint: The process where a device obtains the cryptographic key material to identify and trust future interactions with a network. This term is taken from Konrad Lorenz’s work in biology with new ducklings: "during a critical period, the duckling would assume that anything that looks like a mother duck is in fact their mother" [Stajano99theresurrecting]. An equivalent for a device is to obtain the fingerprint of the network’s root certification authority certificate. A device that imprints on an attacker suffers a similar fate to a duckling that imprints on a hungry wolf. Imprinting is a term from psychology and ethology, as described in [imprinting].

Join Registrar (and Coordinator): A representative of the domain that is configured, perhaps autonomically, to decide whether a new device is allowed to join the domain. The administrator of the domain interfaces with a join registrar (and Coordinator) to control this process. Typically, a join registrar is "inside" its domain. For simplicity, this document often refers to this as just "registrar".

MASA (Manufacturer Authorized Signing Authority): The entity that, for the purpose of this document, signs the vouchers for a manufacturer’s pledges. In some bootstrapping protocols, the MASA may have an Internet presence and be integral to the bootstrapping process, whereas in other protocols the MASA may be an offline service that has no active role in the bootstrapping process.

Owner: The entity that controls the private key of the "pinned-domain-cert" certificate conveyed by the voucher.

Pledge: The prospective device attempting to find and securely join a domain. When shipped, it only trusts authorized representatives of the manufacturer.

Registrar: See join registrar.

TOFU (Trust on First Use): Where a pledge device makes no security decisions but rather simply trusts the first domain entity it is contacted by. Used similarly to [RFC7435]. This is also known as the "resurrecting duckling" model.

Voucher: A signed statement from the MASA service that indicates to
a pledge the cryptographic identity of the domain it should trust.

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. Survey of Voucher Types

A voucher is a cryptographically protected statement to the pledge device authorizing a zero-touch "imprint" on the join registrar of the domain. The specific information a voucher provides is influenced by the bootstrapping use case.

The voucher can impart the following information to the join registrar and pledge:

Assertion Basis: Indicates the method that protects the imprint (this is distinct from the voucher signature that protects the voucher itself). This might include manufacturer-asserted ownership verification, assured logging operations, or reliance on pledge endpoint behavior such as secure root of trust of measurement. The join registrar might use this information. Only some methods are normatively defined in this document. Other methods are left for future work.

Authentication of Join Registrar: Indicates how the pledge can authenticate the join registrar. This document defines a mechanism to pin the domain certificate. Pinning a symmetric key, a raw key, or "CN-ID" or "DNS-ID" information (as defined in [RFC6125]) is left for future work.

Anti-Replay Protections: Time- or nonce-based information to constrain the voucher to time periods or bootstrap attempts.

A number of bootstrapping scenarios can be met using differing combinations of this information. All scenarios address the primary threat of a Man-in-The-Middle (MiTM) registrar gaining control over the pledge device. The following combinations are "types" of vouchers:
<table>
<thead>
<tr>
<th>Voucher Type</th>
<th>Assertion</th>
<th>Registrar ID</th>
<th>Validity</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Log-Veri-</td>
<td>Trust</td>
<td>CN-ID or</td>
</tr>
<tr>
<td></td>
<td>ged</td>
<td>Anchor</td>
<td>DNS-ID</td>
</tr>
<tr>
<td>Audit</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Nonceless Audit</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Owner Audit</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Owner ID</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Bearer</td>
<td>X</td>
<td>wildcard</td>
<td>optional</td>
</tr>
</tbody>
</table>

NOTE: All voucher types include a 'pledge ID serial-number' (not shown here for space reasons).

Audit Voucher: An Audit Voucher is named after the logging assertion mechanisms that the registrar then "audits" to enforce local policy. The registrar mitigates a MiTM registrar by auditing that an unknown MiTM registrar does not appear in the log entries. This does not directly prevent the MiTM but provides a response mechanism that ensures the MiTM is unsuccessful. The advantage is that actual ownership knowledge is not required on the MASA service.

Nonceless Audit Voucher: An Audit Voucher without a validity period statement. Fundamentally, it is the same as an Audit Voucher except that it can be issued in advance to support network partitions or to provide a permanent voucher for remote deployments.

Ownership Audit Voucher: An Audit Voucher where the MASA service has verified the registrar as the authorized owner. The MASA service mitigates a MiTM registrar by refusing to generate Audit Vouchers for unauthorized registrars. The registrar uses audit techniques to supplement the MASA. This provides an ideal sharing of policy decisions and enforcement between the vendor and the owner.

Ownership ID Voucher: Named after inclusion of the pledge’s CN-ID or DNS-ID within the voucher. The MASA service mitigates a MiTM registrar by identifying the specific registrar (via WebPKI) authorized to own the pledge.

Bearer Voucher: A Bearer Voucher is named after the inclusion of a
registrar ID wildcard. Because the registrar identity is not indicated, this voucher type must be treated as a secret and protected from exposure as any 'bearer' of the voucher can claim the pledge device. Publishing a nonceless bearer voucher effectively turns the specified pledge into a "TOFU" device with minimal mitigation against MiTM registrars. Bearer vouchers are out of scope.

5. 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.

This document defines a voucher that is a JSON-encoded instance of the YANG module defined in Section 5.3 that has been, by default, CMS signed.

This format is described here as a practical basis for some uses (such as in NETCONF), but more to clearly indicate what vouchers look like in practice. This description also serves to validate the YANG data model.

Future work is expected to define new mappings of the voucher to Concise Binary Object Representation (CBOR) (from JSON) and to change the signature container from CMS to JSON Object Signing and Encryption (JOSE) or CBOR Object Signing and Encryption (COSE). XML or ASN.1 formats are also conceivable.

This document defines a media type and a filename extension for the CMS-encoded JSON type. Future documents on additional formats would define additional media types. Signaling is in the form of a MIME Content-Type, an HTTP Accept: header, or more mundane methods like use of a filename extension when a voucher is transferred on a USB key.

5.1. Tree Diagram

The following tree diagram illustrates a high-level view of a voucher document. The notation used in this diagram is described in [RFC8340]. Each node in the diagram is fully described by the YANG module in Section 5.3. Please review the YANG module for a detailed description of the voucher format.
module: ietf-voucher

grouping voucher-artifact-grouping
  +-- voucher
    +-- created-on                       yang:date-and-time
    +-- expires-on?                      yang:date-and-time
    +-- assertion                        ianavat:voucher-assertion
    +-- serial-number                    string
    +-- idevid-issuer?                   binary
    +-- pinned-domain-cert               binary
    +-- domain-cert-revocation-checks?   boolean
    +-- nonce?                           binary
    +-- last-renewal-date?               yang:date-and-time

5.2. Examples

This section provides voucher examples for illustration purposes. These examples conform to the encoding rules defined in [RFC8259].

The following example illustrates an ephemeral voucher (uses a nonce). The MASA generated this voucher using the ‘logged’ assertion type, knowing that it would be suitable for the pledge making the request.

```
{
  "ietf-voucher:voucher": {
    "created-on": "2016-10-07T19:31:42Z",
    "assertion": "logged",
    "serial-number": "JADA123456789",
    "idevid-issuer": "base64encodedvalue==",
    "pinned-domain-cert": "base64encodedvalue==",
    "nonce": "base64encodedvalue=="
  }
}
```

The following example illustrates a non-ephemeral voucher (no nonce). While the voucher itself expires after two weeks, it presumably can be renewed for up to a year. The MASA generated this voucher using the ‘verified’ assertion type, which should satisfy all pledges.


```json
{
  "ietf-voucher:voucher": {
    "created-on": "2016-10-07T19:31:42Z",
    "expires-on": "2016-10-21T19:31:42Z",
    "assertion": "verified",
    "serial-number": "JADA123456789",
    "idevid-issuer": "base64encodedvalue==",
    "pinned-domain-cert": "base64encodedvalue==",
    "domain-cert-revocation-checks": "true",
    "last-renewal-date": "2017-10-07T19:31:42Z"
  }
}
```

5.3. YANG Module

5.3.1. "iana-voucher-assertion-type" Module

Following is a YANG [RFC7950] module formally describing the voucher's assertion type.

```yml
<CODE BEGINS>
module iana-voucher-assertion-type {
  namespace "urn:ietf:params:xml:ns:yang:iana-voucher-assertion-type";
  prefix ianavat;

  organization "IANA";
  contact "Internet Assigned Numbers Authority
  Postal: ICANN
  12025 Waterfront Drive, Suite 300
  Los Angeles, CA 90094-2536
  United States of America
  Tel: +1 310 301 5800
  <mailto:iana@iana.org>";
  description "This YANG module defines a YANG enumeration type for IANA registered voucher assertion type. This YANG module is maintained by IANA and reflects the 'voucher assertion types' registry. The lastest revision of this YANG module can be obtained from the IANA web site. Request for new enumerations should be made to IANA via email(iana@iana.org).

  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 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.
```

typedef voucher-assertion {
    type enumeration {
        enum verified {
            value 0;
            description "Indicates that the ownership has been positively verified by the MASA (e.g., through sales channel integration).";
        }
        enum logged {
            value 1;
            description "Indicates that the voucher has been issued after minimal verification of ownership or control. The issuance has been logged for detection of potential security issues (e.g., recipients of vouchers might verify for themselves that unexpected vouchers are not in the log). This is similar to unsecured trust-on-first-use principles but with the logging providing a basis for detecting unexpected events.";
        }
        enum proximity {
            value 2;
            description "Indicates that the voucher has been issued after the MASA verified a proximity proof provided by the device and target domain. The issuance has been logged for detection of potential security issues. This is stronger than just logging, because it requires some
verification that the pledge and owner are
in communication but is still dependent on analysis of
the logs to detect unexpected events.";

enum agent-proximity {
value 3;
description
"Indicates that the voucher has been issued after the
MASA...support of asynchronous enrollment in BRSLI";
}

description "Indicates what kind of ownership is being asserted by voucher"

}

<CODE ENDS>

5.3.2. "ietf-voucher" Module

The revised ietf-voucher YANG module imports the typedef defined in
"iana-voucher-assertion-type" YANG module specified in this document.

<CODE BEGINS>
module ietf-voucher {
  yang-version 1.1;
  namespace "urn:ietf:params:xml:ns:yang:ietf-voucher";
  prefix vch;

  import ietf-yang-types {
    prefix yang;
    reference "RFC 6991: Common YANG Data Types";
  }

  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 iana-voucher-assertion-type {
    prefix ianavat;
    reference "RFCZZZZ: Voucher Profile for Bootstrapping Protocols";
  }

  organization
    "IETF ANIMA Working Group";
  contact
    "WG Web: <https://datatracker.ietf.org/wg/anima/">

This module defines the format for a voucher, which is produced by a pledge’s manufacturer or delegate (MASA) to securely assign a pledge to an ‘owner’, so that the pledge may establish a secure connection to the owner’s network infrastructure.


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

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info).

This version of this YANG module is part of RFC 8366; see the RFC itself for full legal notices.

revision 2021-07-04 {
  description
    "updated to support new assertion enumerated type";
  reference "RFC ZZZZ Voucher Profile for Bootstrapping Protocols";
}

// Top-level statement
rc:yang-data voucher-artifact {
  uses voucher-artifact-grouping;
}

// Grouping defined for future augmentations

grouping voucher-artifact-grouping {
  description

"Grouping to allow reuse/extensions in future work."

container voucher {
  description "A voucher assigns a pledge to an owner (pinned-domain-cert).";
  leaf created-on {
    type yang:date-and-time;
    mandatory true;
    description "A value indicating the date this voucher was created. This node is primarily for human consumption and auditing. Future work MAY create verification requirements based on this node.";
  }
  leaf expires-on {
    type yang:date-and-time;
    must 'not(../nonce)';
    description "A value indicating when this voucher expires. The node is optional as not all pledges support expirations, such as pledges lacking a reliable clock.

    If this field exists, then the pledges MUST ensure that the expires-on time has not yet passed. A pledge without an accurate clock cannot meet this requirement.

    The expires-on value MUST NOT exceed the expiration date of any of the listed 'pinned-domain-cert' certificates.";
  }
  leaf assertion {
    type ianavat:voucher-assertion;
    mandatory true;
    description "The assertion is a statement from the MASA regarding how the owner was verified. This statement enables pledges to support more detailed policy checks. Pledges MUST ensure that the assertion provided is acceptable, per local policy, before processing the voucher.";
  }
  leaf serial-number {
    type string;
    mandatory true;
    description "The serial-number of the hardware. When processing a voucher, a pledge MUST ensure that its serial-number matches this value. If no match occurs, then the pledge MUST NOT process this voucher.";
  }
  leaf idevid-issuer {

type binary;
description
"The Authority Key Identifier OCTET STRING (as defined in
Section 4.2.1.1 of RFC 5280) from the pledge’s IDevID
certificate. Optional since some serial-numbers are
already unique within the scope of a MASA.
Inclusion of the statistically unique key identifier
ensures statistically unique identification of the hardware.
When processing a voucher, a pledge MUST ensure that its
IDevID Authority Key Identifier matches this value. If no
match occurs, then the pledge MUST NOT process this voucher.

When issuing a voucher, the MASA MUST ensure that this field
is populated for serial-numbers that are not otherwise unique
within the scope of the MASA."

leaf pinned-domain-cert {
    type binary;
    mandatory true;
    description
    "An X.509 v3 certificate structure, as specified by RFC 5280,
    using Distinguished Encoding Rules (DER) encoding, as defined
    in ITU-T X.690.

    This certificate is used by a pledge to trust a Public Key
    Infrastructure in order to verify a domain certificate
    supplied to the pledge separately by the bootstrapping
    protocol. The domain certificate MUST have this certificate
    somewhere in its chain of certificates. This certificate
    MAY be an end-entity certificate, including a self-signed
    entity.";
    reference
    "RFC 5280:
    Internet X.509 Public Key Infrastructure Certificate
    and Certificate Revocation List (CRL) Profile.
    ITU-T X.690:
    Information technology - ASN.1 encoding rules:
    Specification of Basic Encoding Rules (BER),
    Canonical Encoding Rules (CER) and Distinguished
    Encoding Rules (DER).";
}
leaf domain-cert-revocation-checks {
    type boolean;
    description
    "A processing instruction to the pledge that it MUST (true)
or MUST NOT (false) verify the revocation status for the
pinned domain certificate. If this field is not set, then
normal PKIX behavior applies to validation of the domain
5.4. CMS Format Voucher Artifact

The IETF evolution of PKCS#7 is CMS [RFC5652]. A CMS-signed voucher, the default type, contains a ContentInfo structure with the voucher content. An eContentType of 40 indicates that the content is a JSON-encoded voucher.

The signing structure is a CMS SignedData structure, as specified by Section 5.1 of [RFC5652], encoded using ASN.1 Distinguished Encoding Rules (DER), as specified in ITU-T X.690 [ITU-T.X690.2015].
To facilitate interoperability, Section 8.3 in this document registers the media type "application/voucher-cms+json" and the filename extension ".vcj".

The CMS structure MUST contain a 'signerInfo' structure, as described in Section 5.1 of [RFC5652], containing the signature generated over the content using a private key trusted by the recipient. Normally, the recipient is the pledge and the signer is the MASA. Another possible use could be as a "signed voucher request" format originating from the pledge or registrar toward the MASA. Within this document, the signer is assumed to be the MASA.

Note that Section 5.1 of [RFC5652] includes a discussion about how to validate a CMS object, which is really a PKCS7 object (cmsVersion=1). Intermediate systems (such as the Bootstrapping Remote Secure Key Infrastructures [BRSKI] registrar) that might need to evaluate the voucher in flight MUST be prepared for such an older format. No signaling is necessary, as the manufacturer knows the capabilities of the pledge and will use an appropriate format voucher for each pledge.

The CMS structure SHOULD also contain all of the certificates leading up to and including the signer’s trust anchor certificate known to the recipient. The inclusion of the trust anchor is unusual in many applications, but third parties cannot accurately audit the transaction without it.

The CMS structure MAY also contain revocation objects for any intermediate certificate authorities (CAs) between the voucher issuer and the trust anchor known to the recipient. However, the use of CRLs and other validity mechanisms is discouraged, as the pledge is unlikely to be able to perform online checks and is unlikely to have a trusted clock source. As described below, the use of short-lived vouchers and/or a pledge-provided nonce provides a freshness guarantee.

6. Design Considerations

6.1. Renewals Instead of Revocations

The lifetimes of vouchers may vary. In some bootstrapping protocols, the vouchers may be created and consumed immediately, whereas in other bootstrapping solutions, there may be a significant time delay between when a voucher is created and when it is consumed. In cases when there is a time delay, there is a need for the pledge to ensure that the assertions made when the voucher was created are still valid.
A revocation artifact is generally used to verify the continued validity of an assertion such as a PKIX certificate, web token, or a "voucher". With this approach, a potentially long-lived assertion is paired with a reasonably fresh revocation status check to ensure that the assertion is still valid. However, this approach increases solution complexity, as it introduces the need for additional protocols and code paths to distribute and process the revocations.

Addressing the shortcomings of revocations, this document recommends instead the use of lightweight renewals of short-lived non-revocable vouchers. That is, rather than issue a long-lived voucher, where the 'expires-on' leaf is set to some distant date, the expectation is for the MASA to instead issue a short-lived voucher, where the 'expires-on' leaf is set to a relatively near date, along with a promise (reflected in the 'last-renewal-date' field) to reissue the voucher again when needed. Importantly, while issuing the initial voucher may incur heavyweight verification checks ("Are you who you say you are?" "Does the pledge actually belong to you?"), reissuing the voucher should be a lightweight process, as it ostensibly only updates the voucher’s validity period. With this approach, there is only the one artifact, and only one code path is needed to process it; there is no possibility of a pledge choosing to skip the revocation status check because, for instance, the OCSP Responder is not reachable.

While this document recommends issuing short-lived vouchers, the voucher artifact does not restrict the ability to create long-lived vouchers, if required; however, no revocation method is described.

Note that a voucher may be signed by a chain of intermediate CAs leading up to the trust anchor certificate known by the pledge. Even though the voucher itself is not revocable, it may still be revoked, per se, if one of the intermediate CA certificates is revoked.

6.2. Voucher Per Pledge

The solution described herein originally enabled a single voucher to apply to many pledges, using lists of regular expressions to represent ranges of serial-numbers. However, it was determined that blocking the renewal of a voucher that applied to many devices would be excessive when only the ownership for a single pledge needed to be blocked. Thus, the voucher format now only supports a single serial-number to be listed.

7. Security Considerations
7.1. Clock Sensitivity

An attacker could use an expired voucher to gain control over a device that has no understanding of time. The device cannot trust NTP as a time reference, as an attacker could control the NTP stream.

There are three things to defend against this: 1) devices are required to verify that the expires-on field has not yet passed, 2) devices without access to time can use nonces to get ephemeral vouchers, and 3) vouchers without expiration times may be used, which will appear in the audit log, informing the security decision.

This document defines a voucher format that contains time values for expirations, which require an accurate clock in order to be processed correctly. Vendors planning on issuing vouchers with expiration values must ensure that devices have an accurate clock when shipped from manufacturing facilities and take steps to prevent clock tampering. If it is not possible to ensure clock accuracy, then vouchers with expirations should not be issued.

7.2. Protect Voucher PKI in HSM

Pursuant the recommendation made in Section 6.1 for the MASA to be deployed as an online voucher signing service, it is RECOMMENDED that the MASA’s private key used for signing vouchers is protected by a hardware security module (HSM).

7.3. Test Domain Certificate Validity When Signing

If a domain certificate is compromised, then any outstanding vouchers for that domain could be used by the attacker. The domain administrator is clearly expected to initiate revocation of any domain identity certificates (as is normal in PKI solutions).

Similarly, they are expected to contact the MASA to indicate that an outstanding (presumably short lifetime) voucher should be blocked from automated renewal. Protocols for voucher distribution are RECOMMENDED to check for revocation of domain identity certificates before the signing of vouchers.

7.4. YANG Module Security Considerations

The YANG module specified in this document defines the schema for data that is subsequently encapsulated by a CMS signed-data content type, as described in Section 5 of [RFC5652]. As such, all of the YANG modeled data is protected from modification.
Implementations should be aware that the signed data is only protected from external modification; the data is still visible. This potential disclosure of information doesn’t affect security so much as privacy. In particular, adversaries can glean information such as which devices belong to which organizations and which CRL Distribution Point and/or OCSP Responder URLs are accessed to validate the vouchers. When privacy is important, the CMS signed-data content type SHOULD be encrypted, either by conveying it via a mutually authenticated secure transport protocol (e.g., TLS [RFC5246]) or by encapsulating the signed-data content type with an enveloped-data content type (Section 6 of [RFC5652]), though details for how to do this are outside the scope of this document.

The use of YANG to define data structures, via the ‘yang-data’ statement, is relatively new and distinct from the traditional use of YANG to define an API accessed by network management protocols such as NETCONF [RFC6241] and RESTCONF [RFC8040]. For this reason, these guidelines do not follow template described by Section 3.7 of [YANG-GUIDE].

8. IANA Considerations

This section deals with actions and processes necessary for IANA to undertake to maintain the "iana-voucher-assertion-type" YANG module. The iana-voucher-assertion-type YANG module is intended to reflect the "voucher assertion types" registry in [TBD].

IANA is asked to create the "iana-voucher-assertion-type YANG module" registry.

Voucher assertion types must not be directly added to the iana-voucher-type YANG module. They must instead be added to the "voucher assertion types" registry.

Whenever a new enumerated type is added to the "voucher assertion types" registry, IANA must also update the "ietf-voucher-assertion-type" YANG module and add a new "enum" statement to the "voucher-assertion-type" type. The assigned name defined by the "enum" statement SHALL be the same as the mnemonic name of the new assertion type. The following substatements to the "enum" statement SHALL be defined:

"value": Use the decimal value from the registry.

"status": Include only if a class or type registration has been deprecated or obsoleted. IANA "deprecated" maps to YANG status "deprecated", and IANA "obsolete" maps to YANG status "obsolete".
"description": Replicate the corresponding information from the registry, namely the full name of the new assertion type.

"reference": Replicate the reference(s) from the registry.

Each time the "iana-voucher-assertion-type" YANG module is updated, a new "revision" statement SHALL be added before the existing "revision" statements. IANA has added this note to the "voucher assertion types" registries:

When this registry is modified, the YANG module "iana-voucher-assertion-type" must be updated as defined in [RFCXXXX]. The "Reference" text in the "voucher assertion types" registry has been updated as follows: OLD: | [RFC8366] NEW: | [RFC8366][RFCXXX]

8.1. The IETF XML Registry

This document registers two URIs in the "IETF XML Registry" [RFC3688].

IANA has registered the following:

Registrant Contact: The ANIMA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

IANA is asked to register a second URI as follows:

Registrant Contact: The ANIMA WG of the IETF.
XML: N/A, the requested URI is an XML namespace.

8.2. The YANG Module Names Registry

This document registers two YANG module in the "YANG Module Names" registry [RFC6020].

IANA is asked to registrar the following:

name: ietf-voucher
prefix: vch
reference: :RFC 8366

IANA is asked to register a second YANG module as follows:

name: iana-voucher-assertion-type
8.3. The Media Types Registry

This document requests IANA to update the following "Media Types" entry to point to the RFC number that will be assigned to this document:

Type name: application
Subtype name: voucher-cms+json
Required parameters: none
Optional parameters: none
Encoding considerations: CMS-signed JSON vouchers are ASN.1/DER encoded.
Security considerations: See Section 7
Interoperability considerations: The format is designed to be broadly interoperable.
Published specification: RFC 8366
Applications that use this media type: ANIMA, 6tisch, and NETCONF zero-touch imprinting systems.
Fragment identifier considerations: none
Additional information: Deprecated alias names for this type: none
Magic number(s): None
File extension(s): .vcj
Macintosh file type code(s): none
Person and email address to contact for further information: IETF ANIMA WG
Intended usage: LIMITED
Restrictions on usage: NONE
8.4. The SMI Security for S/MIME CMS Content Type Registry

This document requests IANA to update this registered OID in the "SMI Security for S/MIME CMS Content Type (1.2.840.113549.1.9.16.1)" registry to point to the RFC number to be assigned to this document:

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Description</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>id-ct-animaJSONVoucher</td>
<td>RFC 8366</td>
</tr>
</tbody>
</table>

Table 1

9. References

9.1. Normative References


9.2. Informative References


Acknowledgements

The authors would like to thank for following for lively discussions on list and in the halls (ordered by last name): William Atwood, Toerless Eckert, and Sheng Jiang.
Russ Housley provided the upgrade from PKCS7 to CMS (RFC 5652) along with the detailed CMS structure diagram.

Authors’ Addresses

Kent Watsen
Juniper Networks
Email: kwatsen@juniper.net

Michael C. Richardson
Sandelman Software
Email: mcr+ietf@sandelman.ca
URI: http://www.sandelman.ca/

Max Pritikin
Cisco Systems
Email: pritikin@cisco.com

Toerless Eckert
Futurewei Technologies Inc.
2330 Central Expy
Santa Clara, 95050
United States of America
Email: tte+ietf@cs.fau.de

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing
210012
China
Email: maqiufang1@huawei.com
Autonomic IP Address To Access Control Group ID Mapping

draft-yizhou-anima-ip-to-access-control-groups-02

Abstract

This document defines the autonomic technical Objectives for IP address/prefix to access control group IDs mapping information. 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 facilitate the group based policy enforcement.

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 May 19, 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
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 ID mapping information of user in order to execute the group based policy. This mapping information is usually 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 access control group ID mappings has to be passed to PEP.
This document defines the autonomic technical Objectives for IP address/prefix and access control group ID 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. A campus policy management system can integrate the function introduced in this document when necessary. Such an Autonomic Service Agent (ASA) performing the function of IP address/prefix to access control group ID mapping is called IPAddressToAccessControlGroups ASA in this document.

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 group IDs. AAP may get the information from one or different resources, for example, DHCP [RFC2131] [RFC8415] server and/or RADIUS [RFC2865] 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 holds. 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 have no change either. For example, the policy can be "Allow the traffic from group engineering whose group ID is 3 to group testing whose group ID is 15", or "assign the traffic destined to VIP group whose group ID is 1 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 extracted and then mapped to the group ID. For instance, when a packet with source address X and destination address Y is received by PEP, PEP checks its mapping table to get that source address X maps to group ID 3 and destination address Y maps to group ID 15. It checks its policy table to see what kind of policy, such as "allow" or "drop", should be enforced on packet from group ID 3 to group ID 15. Then PEP executes the group based policy. The mapping table is short for IP address to access control group ID mapping table. For the information in the mapping table, we call it IP and group mapping information in this document.

IP and group mapping information is usually first available at the access authentication point (AAP). AAP may serve as the DHCP relay which remembers the IP address assigned to the client during DHCP address assignment and at the same time it talks to AAA server to get the client’s group ID information based on client’s identity using AAA protocol such as RADIUS [RFC2865]. AAP then obtains the IP and group mapping information. 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. The problem to be solved by Autonomic Networking Infrastructure (ANI) here is how to make IP address and access control group ID mapping information passing from AAP to PEPs using IPAddressToAccessControlGroups ASA.
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 which are the access nodes for the attaching clients.
Some deployment uses a centralized controller to distribute IP and group ID mapping information. Every single AAP reports its IP and group ID mapping information to the controller. Controller pushes the information regularly to all the PEPs. In addition, when a PEP receives a data packet without pre-stored mapped group ID information.
of the corresponding IP addresses, 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
and/or synchronization 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 ID Mapping Procedures

IPAddressToAccessControlGroups ASA carries out the the function of IP
address/prefix to access control groups ID mapping in this document.
The procedures is illustrated below. As noted in Section 3, a
network node with IPAddressToAccessControlGroups ASA deployed usually
has a role of either AAP or PEP. Therefore two new GRASP Objectives
are defined and used for Objective name based multiplexing. They are
IpToGroupId.PEP and IpToGroupId.AAP respectively. Section 5 gives
more details of the format of them.

The basic procedures are AAP provides the mapping information to PEPs
whenever it obtains new or updated or withdrawn mapping information.
PEPs will then store the information for future policy enforcement
usage. A rare case is that a PEP requests the group ID for a
specific IP address when it finds that information is required but
not locally stored. AAP possessing such mapping information will
reply to this request.
4.1. Behaviours of IP to Group Mapping Information Providing Nodes

IPAddressToAccessControlGroups ASA with mapping information providing feature is usually an AAP supporting IpToGroupId.AAP Objective option. If a PEP would like to provide mapping information as well to the other PEPs, it is logically an AAP in that procedure. Then such PEPs should support both IpToGroupId.PEP and IpToGroupId.AAP Objective options in its IPAddressToAccessControlGroups ASA.

AAP obtains the mapping of IP address and group IDs of a user in various ways. For instance, use RADIUS [RFC2865] or CAPWAP [RFC5415] to get the user’s access control group IDs during authentication phase and use DHCP snooping to get the user’s assigned IP address. Therefore the IP and group ID mapping information of a user can be obtained by AAP at the very early stage when the user connects to the network. Sometimes such mapping information can be statically provisioned based on port or VLAN. Mapping information obtained in such ways is stored locally on AAP. AAP discovers the IPAddressToAccessControlGroups ASA supporting IpToGroupId.PEP first. Then AAP sends Request Negotiation message to those PEPs with the mapping information it has. Whenever there is a change or withdrawn of the mapping information, AAP has to send Request Negotiation again to PEPs for updating.

The providing nodes of mapping information are usually at the network edges. The requesting or receiving nodes of the mapping information are normally aggregation or core nodes with more storage and capability to enforce the policy. There are normally only a few of them, for instance two in a campus network. Therefore the number of mapping information receiving nodes is usually much less than the number of providing nodes. Hence it is quite efficient that the information providing AAP nodes proactively send the mapping information to the receiving PEP nodes. It is the most common case how the mapping information is distributed.

In some rare cases that an AAP receives the Request Synchronization with specific IP address and NULL (represented by zero) group ID, it should reply with Synchronization message with the mapped group ID of the specific IP address. If an AAP has no such mapped information available locally, it can reply with an Invalid message.

4.2. Behaviours of IP to Group Mapping Information Receiving or Requesting Nodes

IPAddressToAccessControlGroups ASA with mapping information requesting or receiving feature is usually a PEP supporting IpToGroupId.PEP Objective option. PEPs need to map the IP address/
prefix of the received data packets to one or more group IDs in order to enforce the group based policy.

PEPs deployed IPAddressToAccessControlGroups ASA supporting IpToGroupId.PEP Objective option will receive the Request Negotiation GRASP message with the mapping information from the information providing AAP nodes as shown in Section 4.1. It should save the mapping information locally. And reply with an Negotiation End GRASP message with an Accept option.

It makes the mapping information of the specific IP addresses received and pre-stored in most cases by PEP before the data packet with those addresses as source or destination is received.

However there are cases that the mapped group ID information of the IP address is not pre-stored when a data packet with that IP address arrives, for example due to timeout or unintentional withdrawn of the mapping information. Then PEPs will send the Request Synchronization with the specific IP address and NULL group ID to ask AAPs for the mapping information.

The request can be triggered by the first data packet of a flow. Group based policy requires both the source and destination group IDs which are mapped from source and destination IP addresses respectively. 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 group IDs or only one of them should be requested. If the requesting node is a tunnel endpoint, usually the inner rather than outer IP addresses should be used to request for the corresponding group id.

The request can also be sent periodically or voluntarily. It can be sent when a newly booted requesting node wants to get the whole set of mapping information or when a requesting node would like have an explicit refreshment on some specific information.

The requesting PEP should send out a GRASP Discovery message containing IpToGroupId.AAP Objective option in order to discover AAPs. It then acts as a GRASP synchronization initiator by sending the Request Synchronization with IP address and NULL group ID as the Objective values to ask for the mapping information. This starts a GRASP synchronization process.
5. Autonomic IP Address to Access Control Groups Objectives

This section defines two GRASP technical Objective options IpToGroupId.AAP and IpToGroupId.PEP that can be used by IPAddressToAccessControlGroups ASA to support autonomic IP address/prefix to access control group ID mapping information distribution.

5.1. IpToGroupId.AAP and IpToGroupId.PEP Objective Option

Both IpToGroupId.AAP and IpToGroupId.PEP Objective option are GRASP Objective options conforming to [RFC8990]. They share the same Objective option value format defined in this section. Normally IpToGroupId.AAP Objective option should be supported by IPAddressToAccessControlGroups ASA deployed on AAP nodes to provide the mapping information and IpToGroupId.PEP Objective option should be supported by IPAddressToAccessControlGroups ASA deployed on PEP nodes to request or receive the mapping information.

The Objective carries the IP prefix/address and its mapping access control group IDs. The format of them 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 = ["IpToGroupId.AAP",
    objective-flags, loop-count,
    [ip-address-or-prefix, *group-id]]

objective = ["IpToGroupId.PEP",
    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 = &(
    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 usually uses 16 bits to present a group ID. But
the representation does not limit that. Zero group ID represents a
NULL group value and is used for full retraction of a prefix or
address.
5.2. Example of Using the Defined Objective Options

Figure 1 shows a typical campus network of with three access switches which are AAPs and two core switches which are PEPs. We assume that the policy in this campus is outsource_group (which has group ID 5) is not allowed to access accounting_group (which has group ID 10). The policy (5, 10, drop) expressed in the form of (source group ID, destination group ID, action) is provisioned on the PEPs which are core switches in the figure.

When a user gets connected, the access switch which is an AAP snoops the DHCP address assignment exchange to obtain the IP address IP_A. The user provides a user ID to get authenticated via 802.1x and RADIUS protocol. Thus the access switch obtains the user’s group ID which is 5 in this example in authentication procedures. So the access switch has the mapping information (IP_A, 5) in the form of (IP address, access control group ID). The mapping information is then passed from the access switch to the core switches which are PEPs using GRASP Objective defined in this document. Figure 3 shows an example of the procedures. Only the key values of the Objective is shown for simplicity.
Figure 3: Example of AAP pushing mapping information to PEPs

After the core switches get this mapping information, they save it for future policy enforcement. For example, when a data packet with source IP address IP_A and destination IP address IP_B is received, the PEP checks its mapping table to get the group ID 5 for IP_A and group ID 10 for IP_B. Then the policy provisioned as (5, 10, drop) is enforcement. So the data packet will be dropped. It facilitates the group based policy execution.
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 two new GRASP Objective option names: "IpToGroupId.AAP" and "IpToGroupId.PEP". The IANA is requested to added them 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


9.2. Informative References


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.

"IpToGroupId.PEP", 15, 101,
[54([4, h'5A0386A78BA56FA4BABC734281C51']], 3506, 2698, 4562]

"IpToGroupId.PEP", 5, 73, [52(h'9946B8A3'), 2881, 2265, 1720, 2450]

"IpToGroupId.PEP", 15, 161,
[54(h'39F3045B641AD291B057CD1857A7314A')]

"IpToGroupId.PEP", 15, 2, [52(h'98A1CE4F')]

"IpToGroupId.PEP", 15, 66, [52(h'69A16BFE'), 2601, 1851, 3876, 1405]

"IpToGroupId.AAP", 15, 254,
[54(h'38AB303B8895DC95066CE00248D2FE91'), 4019, 1166, 3113]

"IpToGroupId.AAP", 15, 63, [52([4, h'0B48']), 3035, 1181]

"IpToGroupId.AAP", 15, 44, [52(h'01F1D8FF'), 3099, 1577, 1139, 1670]

"IpToGroupId.AAP", 15, 181,
[54(h'2C74719F9355BA4E3BDE5689D1FE4CB0')]

"IpToGroupId.PEP", 15, 129, [52(h'AF97C7'), 3149, 2728]

Authors’ Addresses

Yizhou Li
Huawei Technologies

Email: liyizhou@huawei.com
Li Shen
Huawei Technologies

Email: kevin.shenli@huawei.com

Yujing Zhou
Huawei Technologies

Email: zhouyujing3@huawei.com