

ANIMA WG
Internet-Draft
Intended status: Standards Track
Expires: 31 October 2022

S. Fries
T. Werner
Siemens
E. Lear
Cisco Systems
M. Richardson
Sandelman Software Works
29 April 2022

BRSKI with Pledge in Responder Mode (BRSKI-PRM)
draft-ietf-anima-brski-prm-03

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.

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 31 October 2022.

Internet-Draft

BRSKI-PRM

April 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	4
3.	Scope of Solution	5
3.1.	Supported Environment	6
3.2.	Application Examples	6
3.2.1.	Building Automation	6
3.2.2.	Infrastructure Isolation Policy	7
3.2.3.	Less Operational Security in the Target-Domain	7
3.3.	Limitations	7
4.	Requirements Discussion and Mapping to Solution-Elements	7
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	17
5.4.2.	Discovery of Pledge by Registrar-Agent	17
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	38
5.5.4.	Telemetry status handling (registrar-agent - domain registrar)	42
6.	Artifacts	44

6.1.	Voucher Request Artifact	44
6.1.1.	Tree Diagram	44
6.1.2.	YANG Module	44
7.	IANA Considerations	48
7.1.	BRSKI .well-known Registry	48

8.	Privacy Considerations	49
9.	Security Considerations	49
9.1.	Exhaustion Attack on Pledge	49
9.2.	Misuse of acquired Voucher and Enrollment objects by Registrar-Agent	49
9.3.	Misuse of Registrar-Agent Credentials	50
9.4.	YANG Module Security Considerations	50
10.	Acknowledgments	50
11.	References	50
11.1.	Normative References	51
11.2.	Informative References	52
Appendix A.	History of Changes [RFC Editor: please delete] . . .	53
	Authors' Addresses	59

[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 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. As the described solution will rely on additional wrapping signature it will require pre-processing specifically for EST. EST simpleenroll uses PKCS#10 requests only.

[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

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

POP: Proof of possession (of a private key)

POI: Proof of identity

PVR: Pledge-voucher-request

RA: Registration authority, an optional system component to which a CA delegates certificate management functions such as authorization checks.

RER: Registrar-enrollment-request

RVR: Registrar-voucher-request

[3.](#) Scope of Solution

[3.1.](#) Supported Environment

The described solution is applicable in environments in which pledges have a different technology stack or pledges have no direct connection to the domain registrar, but are expected to be managed by this registrar. This can be motivated by pledges deployed in networks not connected to the operational customer site/domain, e.g., during construction of a site. Another application is the preparation of cabinets, which are to be prepared to be installed on a customer site/domain. As there is no direct connection to the registrar available in these environments the solution specified in this document lets the pledges act in a server role so that they can be accessed by a commissioning tool to trigger the bootstrapping. As BRSKI focuses on the pledge in a client role, initiating the bootstrapping, this document defines pledges acting as a server

answering to requests for pledge-voucher-request objects and certification objects.

[3.2.](#) Application Examples

The following examples are intended to motivate the support of additional bootstrapping approaches. Industrial application use cases are introduced, which could leverage BRSKI as such but additionally require to support pledges acting as server only responding to requests, as well as scenarios with limited connectivity to the registrar.

[3.2.1.](#) 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.

[3.2.2.](#) 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.

[3.2.3.](#) 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 forum currently 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.

[3.3.](#) Limitations

The mechanisms in this draft 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 3.2](#) 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 and enrollment request objects are then 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 be 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. 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](#)]. A general goal for the registrar-agent implementation is the reuse of already defined endpoints of the domain registrar. The already existing registrar endpoints have been enhanced to provide distinct endpoints for providing objects with additional signatures for the enrollment objects in [Section 5.3](#).

Internet-Draft

BRSKI-PRM

April 2022

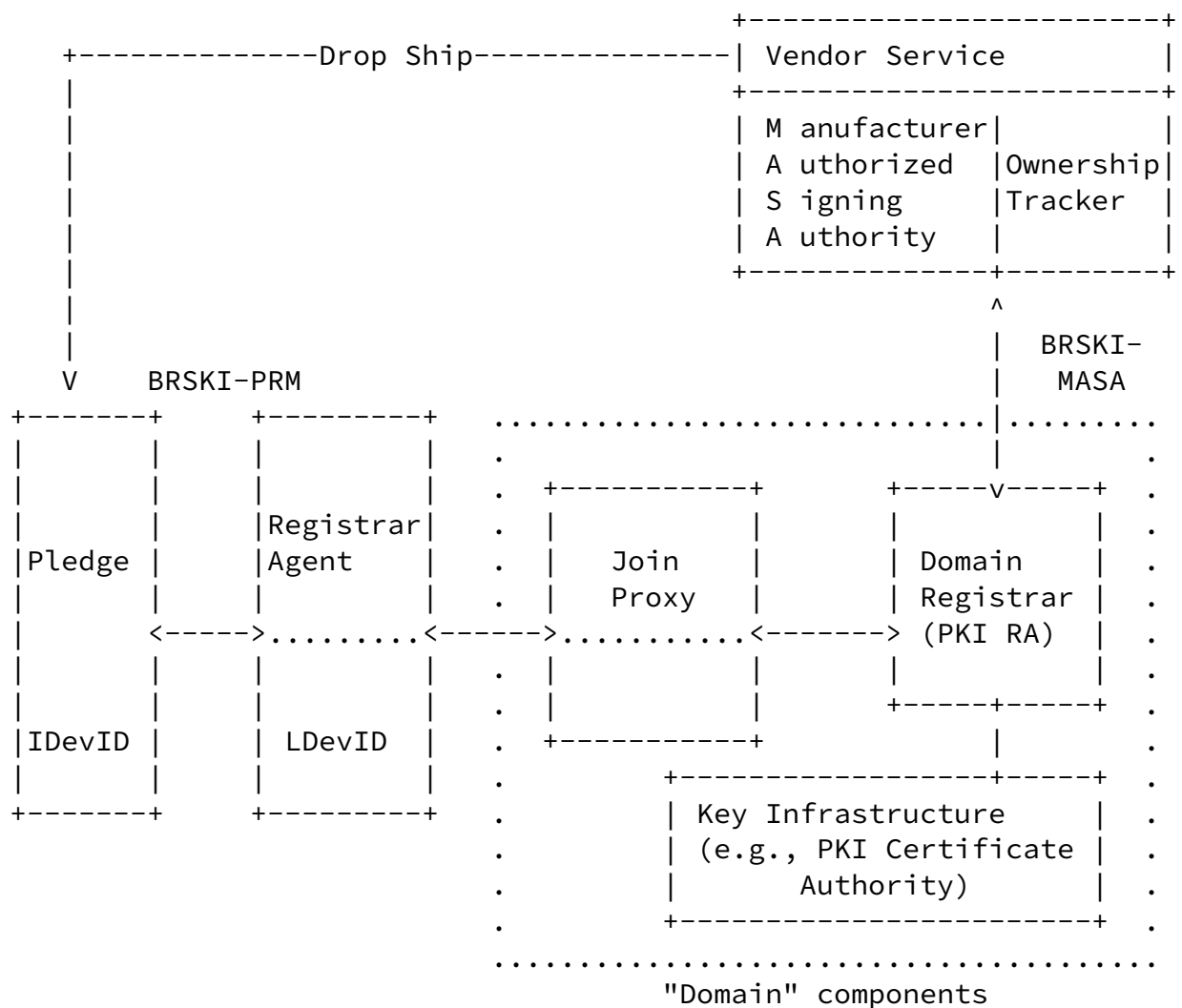


Figure 1: Architecture overview using registrar-agent

For authentication to the domain registrar, the registrar-agent uses

its LDevID(RegAgt). The provisioning of the registrar-agent LDevID may be done by a separate BRSKI run or other means in advance. 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:

- * 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 domain registrar by the pledge is not needed as the pledge will be triggered by the registrar-agent. This enables the registrar to verify that the pledge was contacted by an authorized registrar. In addition, it enables the MASA to provide an agent-proximity assertion.
 - 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 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, pledge-voucher-request objects and pledge-enrollment-request 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: same functionality as described in [\[RFC8995\]](#). Note that it may be used by the registrar-agent instead of the pledge to find the registrar, if not configured.
- * 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 and not directly to the pledge as defined in [Section 5.5](#). 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) EE certificate and has no proof-of-possession of the corresponding private key for the certificate. The pledge 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) EE certificate can be verified after the voucher has been processed by the pledge through a verification of an additional signature of the returned voucher by the registrar if contained (optional feature).

[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 pledge-voucher-request and pledge-enrollment-request 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:

Operation	Operation path	Details
-----------	----------------	---------

Trigger pledge-voucher- request creation	/pledge-voucher-request	Section 5.5.1
Returns pledge-voucher-request		
+-----+		
Trigger pledge- enrollment-request	/pledge-enrollment-request	Section 5.5.1
Returns pledge- enrollment-request		
+-----+		
Provide voucher to pledge	/pledge-voucher	Section 5.5.3
Returns pledge-voucher-status		
+-----+		
Provide enrollment response to pledge	/pledge-enrollment	Section 5.5.3
Returns pledge- enrollment-status		
+-----+		
Provide CA certs to pledge (OPTIONAL)	/pledge-CACerts	
+-----+		

Figure 2: Endpoints on the pledge

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) EE 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, like scanning of product-serial-number information using a QR code or similar.

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 at "product-serial-number.brski-pledge._tcp.local." 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." It can then be discovered by the registrar-agent via mDNS. Note that other mechanisms for discovery may be used.

The registrar-agent is able to build the same information based on the provided list of product-serial-number.

[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. 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) EE certificate) to the pledge to be included into the "agent-provided-proximity-registrar-certificate" leaf of the pledge-voucher-request 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 pledge-voucher-request. Optionally the registrar-agent may provide its LDevID(RegAgt) EE certificate (and optionally also the issuing CA certificate) to the

pledge to be used in the "agent-sign-cert" component of the pledge-voucher-request. If contained, the LDevID(RegAgt) EE 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-agent MUST fetch the LDevID(RegAgt) EE certificate based on the SubjectKeyIdentifier (SKID) in the header of the agent-signed-data of the pledge-voucher-request. The registrar includes the LDevID(RegAgt) EE certificate information into the registrar-voucher-request if the pledge-voucher-requests contains the assertion of "agent-proximity".

The MASA in turn verifies the LDevID(Reg) EE certificate is included in the pledge-voucher-request (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) EE certificate information is contained in the "agent-sign-cert" component of the registrar-voucher-request, 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". 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.

+-----+	+-----+	+-----+	+-----+	+-----+
Pledge	Registrar	Domain	Domain	Vendor
	Agent	Registrar	CA	Service

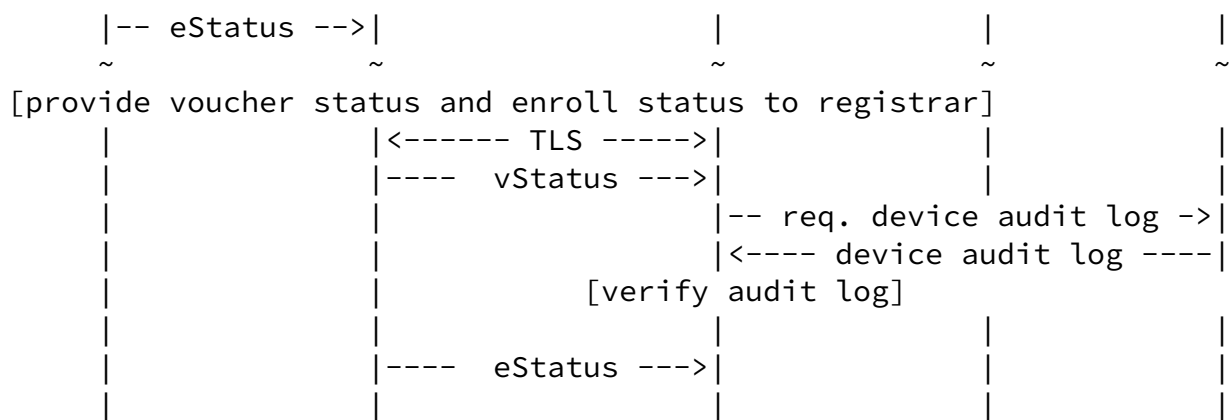


Figure 3: Overview pledge-responder-mode exchanges

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.

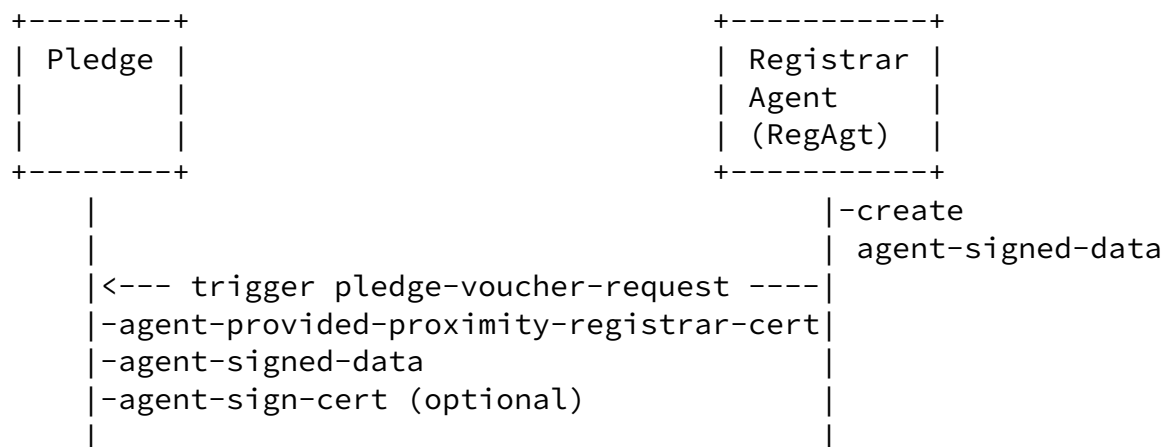
[5.5.1](#). Request Objects Acquisition by Registrar-Agent from Pledge

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 MAY be configured with the product-serial-number(s) of the pledge(s) to be bootstrapped. Note that the product-serial-number may have been used during the pledge discovery already.
- * Registrar: possesses/trusts IDevID CA certificate and an own LDevID(Reg) credential.
- * MASA: possesses own credentials (voucher signing key, TLS server certificate) as well as IDevID CA certificate of pledge vendor / manufacturer and site-specific LDevID CA certificate.



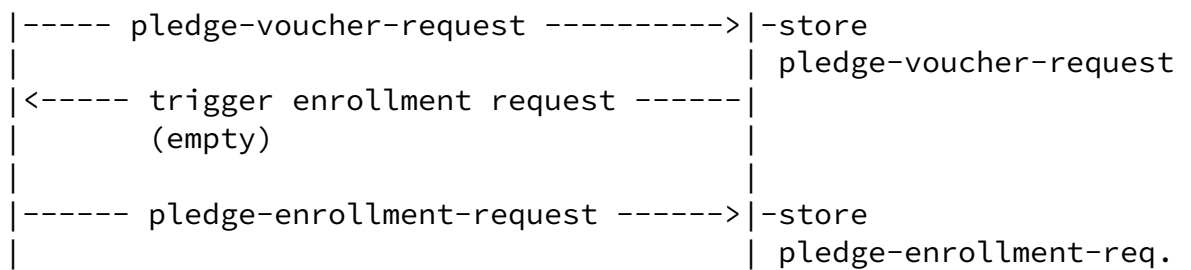


Figure 4: Request collection (registrar-agent - pledge)

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

The registrar-agent pledge-voucher-request Content-Type header is: `application/json`. It defines a JSON document to provide three parameter:

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

The the trigger for the pledge to create a pledge-voucher-request is depicted in the following figure:

```

{
  "agent-provided-proximity-registrar-cert": "base64encodedvalue==",
  "agent-signed-data": "base64encodedvalue==",
  "agent-sign-cert": ["base64encodedvalue==", "base64encodedvalue==", "..."]
}
  
```

Figure 5: Representation of trigger to create pledge-voucher-request

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) EE

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: contains the base64-encoded SubjectKeyIdentifier 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.

```

{
  "payload": {
    "ietf-voucher-request-prm:agent-signed-data": {
      "created-on": "2021-04-16T00:00:01.000Z",
      "serial-number": "callee4711"
    },
    "signatures": [
      {
        "protected": {
          "alg": "ES256",
          "kid": "base64encodedvalue=="
        },
        "signature": "base64encodedvalue=="
      }
    ]
  }
}

```

Figure 6: Representation of agent-signed-data

Upon receiving the voucher-request trigger, the pledge SHALL construct the body of the pledge-voucher-request 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 pledge-voucher-request it SHOULD respond with HTTP 406 error code to the registrar-agent to indicate that it is not able to create the pledge-voucher-request.

The header of the pledge-voucher-request 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 pledge-voucher-request (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.

Internet-Draft

BRSKI-PRM

April 2022

- * `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)` EE 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 base64encoded certificate information. It starts from the base64-encoded `LDevID(RegAgt)` EE certificate optionally followed by the issuing CA certificate and potential further certificates. If supported, it MUST at least contain the `LDevID(RegAgt)` EE certificate provided as trigger parameter.

The enhancements of the YANG module for the `ietf-voucher-request` with these new leafs are defined in [Section 6.1](#).

The object is signed using the pledge's `IDevID` credential contained as `x5c` parameter of the JOSE header.

Internet-Draft

BRSKI-PRM

April 2022

```
{
  "payload": {
    "ietf-voucher-request-prm:voucher": {
      "created-on": "2021-04-16T00:00:02.000Z",
      "nonce": "eDs++/FuDHGUnRxN3E14CQ==",
      "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 pledge-voucher-request

The pledge-voucher-request 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 pledge-voucher-request 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 pledge-enrollment-request 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: initial or re-enroll as shown in Figure 8.

```
{
  "enroll-type" = "intial"
}
```

Figure 8: Example of trigger to create a pledge-enrollment-request

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

Upon receiving the enrollment-trigger, the pledge SHALL construct the pledge-enrollment-request 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 pledge-enrollment-request 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 pledge-enrollment-request

With the collected pledge-voucher-request object and the pledge-enrollment-request object, the registrar-agent starts the interaction with the domain registrar.

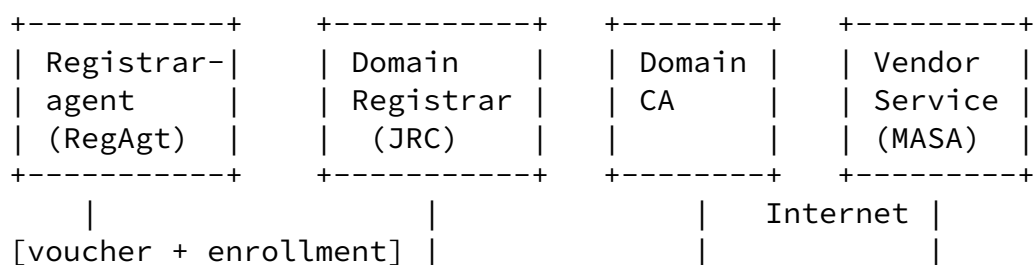
As the registrar-agent is intended to facilitate communication between the pledge and the domain registrar, a collection of requests from more than one pledge is possible, allowing a bulk bootstrapping of multiple pledges using the same connection between the registrar-agent and the domain registrar.

[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/manufacture 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 pledge-voucher-request and pledge-enrollment-request objects(s).
- * Registrar: possesses the IDevID CA certificate of the pledge vendor/manufacture and an it's own LDevID(Reg) credentials of the site domain.
- * MASA: possesses it's own vendor/manufacture credentials (voucher signing key, TLS server certificate) related to pledges IDevID and the site-specific LDevID CA certificate.



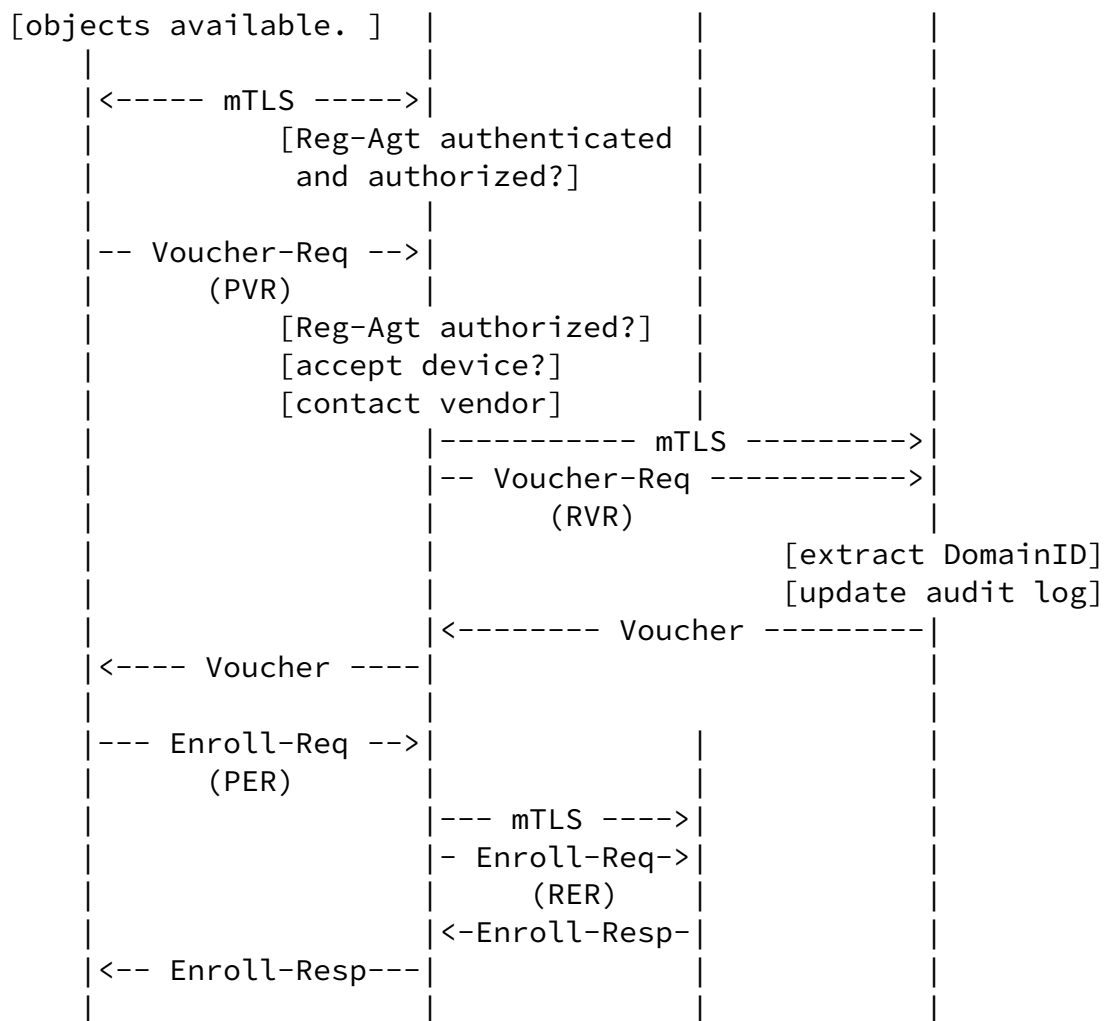


Figure 10: Request processing between registrar-agent and infrastructure bootstrapping services

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 from registrar-agent to registrar is established, the authorization SHALL be verified again based on the agent-signed-data contained in the pledge-voucher-request (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 pledge-voucher-request by HTTP POST to the registrar endpoint: `"/.well-known/brski/requestvoucher"`

The Content-Type header field for JSON-in-JWS pledge-voucher-request 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 pledge-voucher-request 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 pledge-voucher-request (PVR):

- * `agent-provided-proximity-registrar-cert`: MUST contain registrar's own LDevID(Reg) EE 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 pledge-voucher-request, 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) EE certificate. If contained the registrar MUST verify that the LDevID(ReAgt) EE

Internet-Draft

BRSKI-PRM

April 2022

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) EE 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 pledge-voucher-request is unknown.

If the validation succeeds, the registrar SHOULD accept the pledge-voucher-request (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 registrar-voucher-request (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 registrar-voucher-request (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 registrar-voucher-request (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 registrar-voucher-request creation time.

- * nonce: copied from the pledge-voucher-request

- * serial-number: contains the pledge product-serial-number. The registrar MUST verify that the LDevID EE certificate subject serialNumber 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 pledge-voucher-request provided by the registrar-agent.

The registrar-voucher-request (RVR) can be enhanced optionally with the following parameter as defined in [Section 6.1](#):

- * agent-sign-cert: contains the LDevID(RegAgt) EE certificate or the LDevID(RegAgt) EE 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) EE certificate. If multiple certificates are included in the x5c, the first MUST be the base64-encoded LDevID(RegAgt) EE certificate.

The MASA uses this information for verification of the agent is in proximity to the registrar to state the corresponding assertion "agent-proximity". If the agent-sign-cert is not included in the registrar-voucher-request (RVR), it is also contained in the "prior-signed-voucher-request" field carrying the pledge-voucher-request PVR.

The object is signed using the registrar LDevID(Reg) credential,

which corresponds to the certificate signaled in the JOSE header.

```
{
  "payload": {
    "ietf-voucher-request-prm:voucher": {
      "created-on": "2022-01-04T02:37:39.235Z",
      "nonce": "eDs++/FuDHGUnRxN3E14CQ==",
      "serial-number": "callee4711",
      "assertion": "agent-proximity",
      "prior-signed-voucher-request": "base64encodedvalue==",
      "agent-sign-cert": [
        "base64encodedvalue==",
        "base64encodedvalue==",
        "..."
      ]
    },
    "signatures": [
      {
        "protected": {
          "alg": "ES256",
          "x5c": [ "base64encodedvalue==" ]
        },
        "signature": "base64encodedvalue=="
      }
    ]
  }
}
```

Figure 11: Representation of registrar-voucher-request

The registrar SHALL send the registrar-voucher-request (RVR) to the

MASA endpoint by HTTP POST: `"/.well-known/brski/requestvoucher"`

The registrar-voucher-request Content-Type header field is defined in [\[I-D.ietf-anima-jws-voucher\]](#) as: `application/voucher-jws+json`

The registrar-voucher-request (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 registrar-voucher-request (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

registrar-voucher-request (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 registrar-voucher-request (RVR). The LDevID(RegAgt) EE 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 registrar-voucher-request MUST contain the corresponding LDevID(RegAgt) certificate data in the "agent-sign-cert" field of either the LDevID(RegAgt) EE certificate of registrar-voucher-request (RVR) or of pledge-voucher-request (PVR) from "prior-signed-voucher-request" field. It can be verified by the MASA that it was issued by the same domain CA as the LDevID(Reg) EE 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) EE 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 is indicated by the Accept header field of the RVR or MASA SHOULD respond with the same format as the PVR was (default "JSON-in-JWS") Specifically 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.

```
{
  "payload": {
    "ietf-voucher:voucher": {
      "assertion": "agent-proximity",
      "serial-number": "callee4711",
      "nonce": "eDs++/FuDHGUnRxN3E14CQ==",
      "created-on": "2022-01-04T00:00:02.000Z",
      "pinned-domain-cert": "MIIBpDCCA...w=="
    },
    "signatures": [
      {
        "protected": {
          "alg": "ES256",
          "x5c": [ "base64encodedvalue==" ]
        },

```

```

        "signature": "base64encodedvalue=="
      }
    ]
  }
}

```

Figure 12: Representation of MASA issued 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 MAY 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 pledge-voucher-request in the agent-provided-proximity-registrar-cert component. Figure Figure 13 below provides an example of the voucher with two signatures.

```

{
  "payload": {
    "ietf-voucher:voucher": {
      "assertion": "agent-proximity",
      "serial-number": "callee4711",
      "nonce": "eDs++/FuDHGUnRxN3E14CQ==",
      "created-on": "2022-01-04T00:00:02.000Z",
      "pinned-domain-cert": "MIIBpDCCA...w=="
    }
  }
}

```

```

    },
    "signatures": [
      {
        "protected": {
          "alg": "ES256",
          "x5c": [ "base64encodedvalue==" ]
        },
        "signature": "base64encodedvalue=="
      },
      {
        "protected": {
          "alg": "ES256",
          "x5c": [ "base64encodedvalue==" ]
        },
        "signature": "base64encodedvalue=="
      }
    ]
  }
}

```

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 pledge-enrollment-request (PER) to the registrar. Deviating from BRSKI the pledge-enrollment-request (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.

/serverkeygen) on the registrar cannot be used for 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) EE certificate in the TLS establishment and the provided pledge-voucher-request (PVR) as JSON-in-JWS object.

- * If the registrar receives a pledge-enrollment-request (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 IDDevID), carried in "x5c" header field, is accepted to join the domain after successful validation of the pledge-voucher-request.
- * 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 enhance the PKCS#10 request or generate a structure containing the attributes to be included by the CA into the requested LDevID EE certificate and sends both (the original PKCS#10 request and the enhancements) to the domain CA. As enhancing the PKCS#10 request destroys the initial proof of possession of the corresponding private key, the CA would need to accept RA-verified requests. This request handling to the domain CA 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"`

If validation of the wrapping signature fails, the registrar SHOULD respond with the HTTP 404 error code. The HTTP 406 error code SHOULD be used, if the pledge-enrollment-request (PER) is in an unknown format.

A situation that could be resolved with administrative action (such as adding a vendor/manufacturer IDevID CA as trusted party) MAY be responded with the HTTP 403 error code.

A successful interaction with the domain CA will result in a pledge LDevID EE certificate, which is then forwarded by the registrar to the registrar-agent using the Content-Type header: application/pkcs7-mime.

The registrar-agent has now finished the exchanges with the domain registrar and can supply the voucher-response (from MASA via Registrar) and the enrollment-response (LDevID EE 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 restart 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.

Internet-Draft

BRSKI-PRM

April 2022

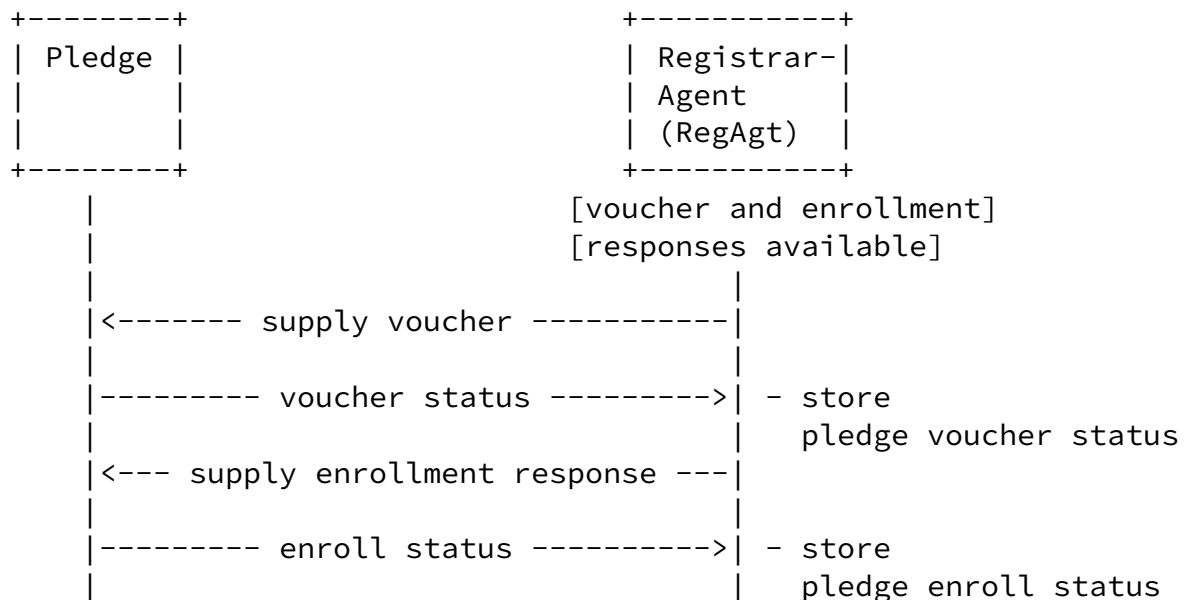


Figure 14: Response and status handling between pledge and registrar-agent

The registrar-agent provides the information via two 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 is 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 \[RFC8995\]](#).

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 voucher request successfully, including revocation state of the certificate, 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 15. 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 15: Representation of pledge voucher status telemetry

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 EE certificate. The pledge MAY omit the revocation check as the EE LDevID certificate was freshly issued.. 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 EE 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 EE certificate successfully it SHALL sign the response using the LDevID of the pledge as shown in Figure 16. 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 16: 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.

+-----+ +-----+ +-----+ +-----+

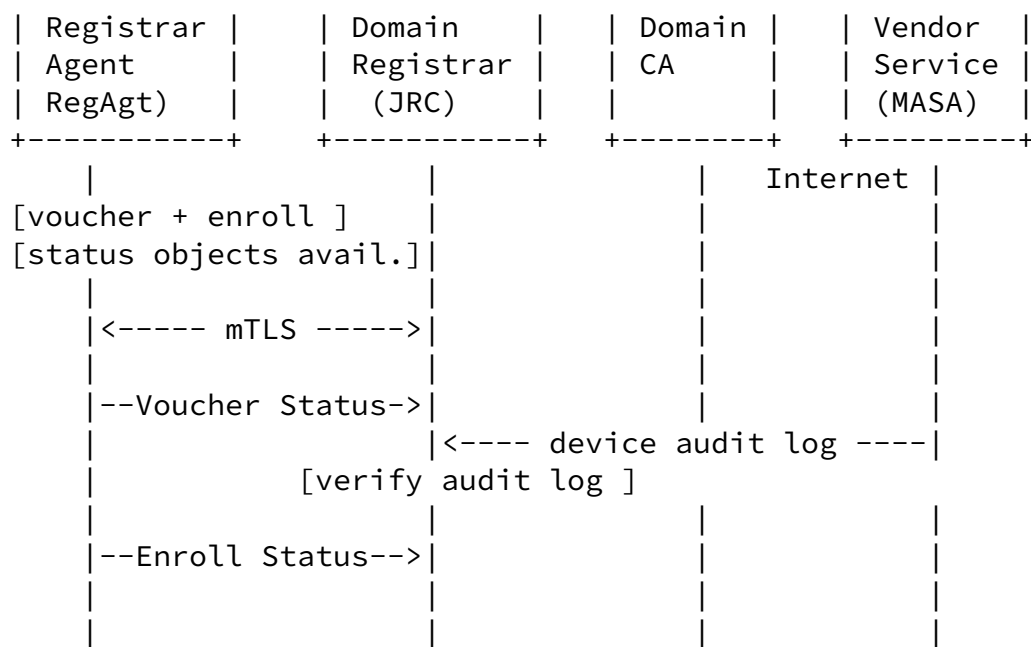


Figure 17: Bootstrapping status handling

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 14 and depicted in the example in Figure 15.

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 but MAY simply fail with an HTTP 404 error. 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 14 and depicted in the example in Figure 16.

The registrar SHALL verify the signature of the pledge enroll status object. In case the enroll status object indicates success 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 EE certificate referenced in the header of the enroll status object. In case the enroll status object indicates a failure, the pledge was unable to verify the received LDevID EE certificate and therefore signed the enroll status objects with its IDDevID 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 but MAY simply fail with an HTTP 404 error. 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 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@2021-12-16.yang"
```

```
module ietf-voucher-request-prm {
```

```
  yang-version 1.1;
```

```
  namespace "urn:ietf:params:xml:ns:yang:ietf-voucher-request-prm";  
  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.
  It 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) 2022 IETF Trust and the persons identified as authors of the code. All rights reserved.

Internet-Draft

BRSKI-PRM

April 2022

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 8995](#); see the RFC itself for full legal notices.";

```
revision 2021-12-16 {
  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/extensions in future work.";
  uses vcr:voucher-request-grouping {
    refine "voucher/expires-on" {
      mandatory false;
      description
        "An expires-on field is not valid in a
        voucher-request, and any occurrence MUST be ignored.";
    }
    refine "voucher/pinned-domain-cert" {
      mandatory false;
      description
        "A pinned-domain-cert field is not valid in a
        voucher-request, and any occurrence MUST be ignored.";
    }
  }
}
```

```

    refine "voucher/last-renewal-date" {
      description
        "A last-renewal-date field is not valid in a
        voucher-request, and any occurrence MUST be ignored.";
    }
    refine "voucher/domain-cert-revocation-checks" {
      description
        "The domain-cert-revocation-checks field is not valid in a

```

```

    voucher-request, and any occurrence MUST be ignored.";
  }
  refine "voucher/assertion" {
    mandatory false;
    description
      "Any assertion included in registrar voucher-requests
      SHOULD be ignored by the MASA.";
  }

  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
    RFC 8446: The Transport Layer Security (TLS)
    Protocol Version 1.3";
}

```

```

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";
    }
}
}
}
}
}
<CODE ENDS>

```

Examples for the pledge-voucher-request 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:

Fries, et al.

Expires 31 October 2022

[Page 48]

Internet-Draft

BRSKI-PRM

April 2022

URI	Description	Reference
pledge-voucher-request	create pledge-voucher-request	[THISRFC]
pledge-enrollment-request	create pledge-enrollment-request	[THISRFC]
pledge-voucher	supply voucher response	[THISRFC]
pledge-enrollment	supply enrollment response	[THISRFC]
pledge-CACerts	supply CA certs to pledge	[THISRFC]
requestenroll	supply PER to registrar	[THISRFC]

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 pledge-voucher-request processing on the domain registrar side. This requires the domain registrar to verify the "proximity-registrar-cert" field in the pledge-voucher-request (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 pledge-voucher-request (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) EE certificate information contained in the pledge-voucher-request (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) EE 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) EE 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 based 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 and Oskar Camenzind, for their input and discussion on use cases and call flows.

[11.](#) References

Fries, et al. Expires 31 October 2022 [Page 50]

Internet-Draft

BRSKI-PRM

April 2022

[11.1.](#) Normative References

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

Richardson, M. and T. Werner, "JWS signed Voucher Artifacts for Bootstrapping Protocols", Work in Progress,

Internet-Draft, [draft-ietf-anima-jws-voucher-03](https://www.ietf.org/archive/id/draft-ietf-anima-jws-voucher-03), 7 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-anima-jws-voucher-03.txt>>.

[I-D.ietf-anima-rfc8366bis]

Watsen, K., Richardson, M. C., Pritikin, M., Eckert, T., and Q. Ma, "A Voucher Artifact for Bootstrapping Protocols", Work in Progress, Internet-Draft, [draft-ietf-anima-rfc8366bis-00](https://www.ietf.org/archive/id/draft-ietf-anima-rfc8366bis-00), 31 January 2022, <<https://www.ietf.org/archive/id/draft-ietf-anima-rfc8366bis-00.txt>>.

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

Watsen, K., Housley, R., and S. Turner, "Conveying a Certificate Signing Request (CSR) in a Secure Zero Touch Provisioning (SZTP) Bootstrapping Request", Work in Progress, Internet-Draft, [draft-ietf-netconf-sztp-csr-14](https://www.ietf.org/archive/id/draft-ietf-netconf-sztp-csr-14), 2 March 2022, <<https://www.ietf.org/archive/id/draft-ietf-netconf-sztp-csr-14.txt>>.

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

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", [RFC 6241](https://www.rfc-editor.org/info/rfc6241), DOI 10.17487/RFC6241, June 2011, <<https://www.rfc-editor.org/info/rfc6241>>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", [RFC 6762](https://www.rfc-editor.org/info/rfc6762), DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.

[RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", [RFC 6763](https://www.rfc-editor.org/info/rfc6763), DOI 10.17487/RFC6763, February 2013, <<https://www.rfc-editor.org/info/rfc6763>>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", [RFC 7030](https://www.rfc-editor.org/info/rfc7030), DOI 10.17487/RFC7030, October 2013, <<https://www.rfc-editor.org/info/rfc7030>>.

- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", [RFC 7515](#), DOI 10.17487/RFC7515, May 2015, <<https://www.rfc-editor.org/info/rfc7515>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", [RFC 8040](#), DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in [RFC 2119](#) Key Words", [BCP 14](#), [RFC 8174](#), DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8366] Watsen, K., Richardson, M., Pritikin, M., and T. Eckert, "A Voucher Artifact for Bootstrapping Protocols", [RFC 8366](#), DOI 10.17487/RFC8366, May 2018, <<https://www.rfc-editor.org/info/rfc8366>>.
- [RFC8407] Bierman, A., "Guidelines for Authors and Reviewers of Documents Containing YANG Data Models", [BCP 216](#), [RFC 8407](#), DOI 10.17487/RFC8407, October 2018, <<https://www.rfc-editor.org/info/rfc8407>>.
- [RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", [RFC 8995](#), DOI 10.17487/RFC8995, May 2021, <<https://www.rfc-editor.org/info/rfc8995>>.

11.2. Informative References

- [BRSKI-PRM-abstract]
"Abstract BRSKI-PRM Protocol Overview", April 2022, <https://raw.githubusercontent.com/anima-wg/anima-brski-prm/main/pics/brski_prm_overview.png>.
- [IEEE-802.1AR]
Institute of Electrical and Electronics Engineers, "IEEE 802.1AR Secure Device Identifier", IEEE 802.1AR, June 2018.
- [RFC2986] Nystrom, M. and B. Kaliski, "PKCS #10: Certification Request Syntax Specification Version 1.7", [RFC 2986](#), DOI 10.17487/RFC2986, November 2000, <<https://www.rfc-editor.org/info/rfc2986>>.

Internet-Draft

BRSKI-PRM

April 2022

- [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", [RFC 5280](#), DOI 10.17487/RFC5280, May 2008, <<https://www.rfc-editor.org/info/rfc5280>>.
- [RFC8152] Schaad, J., "CBOR Object Signing and Encryption (COSE)", [RFC 8152](#), DOI 10.17487/RFC8152, July 2017, <<https://www.rfc-editor.org/info/rfc8152>>.
- [RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", [BCP 215](#), [RFC 8340](#), DOI 10.17487/RFC8340, March 2018, <<https://www.rfc-editor.org/info/rfc8340>>.

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

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 is 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 EE 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 EE 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)

Fries, et al.

Expires 31 October 2022

[Page 54]

Internet-Draft

BRSKI-PRM

April 2022

- * 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) EE 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:

Fries, et al.

Expires 31 October 2022

[Page 55]

Internet-Draft

BRSKI-PRM

April 2022

- * 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-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-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 3.2](#).
- * 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 BRSKI 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/>

