ANIMA
Internet-Draft
Updates: draft-ietf-anima-brski-prm, rfc6690, rfc9733 (if approved)
Intended status: Standards Track
Expires: 23 April 2026

T. Eckert, Ed.
Futurewei
E. Dijk
IoTconsultancy.nl

20 October 2025

BRSKI discovery and variations
draft-ietf-anima-brski-discovery-09

Abstract

This document specifies how to make BRSKI communications autoconfiguring, extensible and resilient in the face of simultaneous use of different variations of the BRSKI protocol (BRSKI, BRSKI-AE, BRSKI-PRM, constrained BRSKI, stateless constrained BRSKI proxies). This document specifies a data model, IANA registry and BRSKI component procedures to achieve this.

This document does not define any new discovery methods. Instead, its data model allows to signal all current (and future) variations of the BRSKI family of protocols consistently via different existing network discovery mechanisms: DNS-SD, CoAP discovery (CORE-LF) and GRASP. Additional/future discovery mechanisms can also be supported through the IANA registry.

Automatic resiliency and load-sharing are enabled through the use of discovery mechanisms and the provisioning of multiple instances of BRSKI components such as registrars and Join Proxies. This document specifies the procedures to support load-sharing and (fast) failover under failure and recovery of redundant components.

Future proof deployments of BRSKI requires Join Proxies that automatically support any current and future BRSKI variation. This document specifies the procedures how Join Proxies can support this through specific Join Proxy protocol behavior and the use of discovery mechanisms.

The specification for discovery of pledges by their IDevID as introduced by BRSKI-PRM is refined in this document.

Status of This Memo

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

This Internet-Draft will expire on 23 April 2026.

Copyright Notice

Table of Contents

1. Terminology

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

This document relies on the terminology defined in [BRSKI]. The following terms are described partly in addition.

Context: A set of Services for whom the same set of variations applies

IP, IPv4, IPv6: In this document, IP refers to (inclusively) IPv4 or IPv6. If a statement applies to only one of the two versions, then the terms IPv4 or IPv6 are used.

Initiator: A host that is using an IP transport protocol to initiate a connection or transaction to another host called the responder.

Initiator socket: A socket consisting of an initiators IP address, protocol and protocol port number from which it initiates connections or transactions to a responder (typically UDP or TCP).

Objective Name: See Service Name.

Resource Type: See Service Name.

Responder: A host that is using an IP transport protocol to respond to transaction or connection requests from an Initiator.

Responder socket: A socket consisting of a responders IP address, protocol and protocol port number on which it responds to requests of the protocol (typically UDP or TCP).

Role: In this document, functionality of an entity in a variation of BRSKI that can act as a responder and whose supported variations can be discovered. BRSKI roles relevant in this document include Join Registrar, Join Proxy and pledge. The IANA registry defined by this document allows to specify variations for any roles. See also Context.

Socket: The combination of am IP address, an IP protocol that utilizes a port number (such as TCP or UDP) and a port number of that protocol.

Service Name: The name for (a subset of) the functionality/API provided by a discoverable responder socket. This term is inherited from [DNS-SD] but unless otherwise specified also used in this document to apply to any other discovery functionality/ API. The terminology used by other mechanisms typically differs. For example, when [GRASP] is used to discover a responder socket for BRSKI, the Objective Name carries the equivalent to the service name. In [CORE-LF], the Resource Type (rt=) carries the equivalent of the service name.

Type: See Variation Type.

Variation: A combination one one variation choice each for every variation type applicable to the context of one discoverable BRSKI communications. For example, in the context of BRSKI, a variation is one choice for "mode", one choice for "enroll" and once choice for "vformat".

Variation Type: The name for one aspect of a protocol for which two or more choices exist (or may exist in the future), and where the choice can technically be combined orthogonal to other variation types. This document defined the BRSKI variation types "mode", "enroll" and "vformat".

Variation Type Choice: The name for different values that a particular variation type may have. For example, this document does defines the choices "rrm" and "prm" for the BRSKI variation "mode".

ACP: "An Autonomic Control Plane", [ACP].

BRSKI: "Bootstrapping Remote Secure Key Infrastructure", [BRSKI].

BRSKI-AE: "Alternative Enrollment Protocols in [BRSKI]", [BRSKI-AE].

BRSKI-PRM: "[BRSKI] with Pledge in Responder Mode", [BRSKI-PRM].

cBRSKI: "Constrained Bootstrapping Remote Secure Key Infrastructure ([BRSKI])", [cBRSKI].

COAP: "The Constrained Application Protocol (CoAP)", [COAP].

CORE-LF: "Constrained RESTful Environments (CoRE) Link Format", [CORE-LF].

cPROXY: "Constrained Join Proxy for Bootstrapping Protocols", [cPROXY].

DNS-SD: "DNS-Based Service Discovery", [DNS-SD].

EST: "Enrollment over Secure Transport", [EST].

GRASP: "GeneRic Autonomic Signaling Protocol", [GRASP].

GRASP-DNSSD: "DNS-SD Compatible Service Discovery in GeneRic Autonomic Signaling Protocol (GRASP)", [I-D.eckert-anima-grasp-dnssd].

JWS-VOUCHER: "JWS signed Voucher Artifacts for Bootstrapping Protocols", [JWS-VOUCHER].

lwCMP: "Lightweight Certificate Management Protocol (CMP) Profile", [RFC9483].

mDNS: "multicast DNS", [mDNS].

SCEP: "Simple Certificate Enrolment Protocol", [RFC8894].

2. Overview

2.1. Challenges

BRKI was designed to support multi-vendor deployments ideally with zero additional provisioning in the network just to support BRSKI. In recent years, multiple variations of the BRSKI protocol where specified, sich as [BRSKI-PRM], [BRSKI-AE], [cBRSKI] and [cPROXY]; within these documents that are multiple options that need to be supported by all BRSKI entities involved in a BRSKI enrollment, such as pledge, proxy and registrar.

1. Assume for example a registrar from vendor A is deployed in an enterprise network or manufacturing plant to support a variety of pledges from an ecosystem of vendors all using the vendor A registrar, and a particular subset of BRSKI protocol variations. Later, it becomes necessary to also deploy various pledges from a different ecosystem. Vendor B provides a registrar for this ecosystem, but they do use some slight variation of BRSKI. Now the proxies deployed through either the A or B ecosystem discover either registrars A or B and randomnly pick one. And in case they pick the wrong registrar, enrollment for the pledge will fail because the proxy variation of BRSKI is not compatible with the variation(s) supported by the choosen registrar.

   Now the proxy implementations coming either from A or B start to fix up the issue by introducing some proprietary extension to only discover "their" type of registrar. Now a pledge from the A ecosystem can only be enrolled behind an A proxy and a B pledge only behind a B proxy. The network operator now falls into the next trap: It is not possible anymore to have networks where both A and B pledges can be enrolled, because it is physcially not possible or financially feasible to deploy both A and B proxies. Or if they are deployed, pledges would only randomnly pick the right pledge.

2. Some use-case community wants to introduce a new variation aspect, such as introducing a new encoding method for the message exchanges or a different certificate enrollment protocol. But now this aspect can be combined with any possible other aspect of BRSKI. And without appropriate planning upfront this ends up in more chaos of ad-hoc definition every time some ecosystem prefers one specific variation of options.

3. As if variations where not difficult enough, networks may only support one specific autodiscovery protocol, and specific variations did not bother to define how their variation of BRSKI could be discovered by another discovery protocol mechanism. So that variation then invents yet another one-off way to discover its variation in this new type of discover method in this now more important type of networks.

The following sub-sections attempt to more formally describe the different challenges in a technically more precise language.

2.1.1. Signaling BRSKI variation for responder selection.

When an initiator uses a discovery mechanism such as [DNS-SD] to discover an instance of the service that it intends to connect to, it may discover more than one such instance.

For example, BRSKI pledges want to discover Join Proxies or registrars. In the presence of variations of the BRSKI mechanisms that impact interoperability, performance or security, not all discovered instances may support exactly what the initiator needs to achieve interoperability or they may not provide the best desired metric. To support choosing the best interoperable responder, the discovery mechanism needs to carry the necessary additional information beside the service name that indicates the service/role of the responder.

2.1.2. Consistent support for variations across different discovery mechanisms.

Different BRSKI deployments may prefer different discovery mechanisms, such as [DNS-SD], [GRASP], [CORE-LF] or others. Any variation in discovery already defined for one discovery mechanism usually has to be re-specified individually for every other discovery mechanism. This makes it often cumbersome to select the preferred discovery mechanism for a specific type of deployment, because such additional specification work can take a long time. Indepedent specification of variations for different discovery mechanisms can also easily lead to inconsistencies and hence the inability to equally support all variations across all discovery mechanisms.

2.1.3. Variation agnostic support for Join Proxies

Pledges or Agents often need to connect to a registrar that supports the variation of BRSKI supported by the pledge or Agent via a Join Proxy. The Join Proxy needs to discover registrars and the variations they support and then announce themselves to pledges or Agents accordingly so that when the pledge or Agent connects to the Proxy that it will connect it to the right registrar.

This document defines variations so that Join Proxies can be implemented and operated agnostic of variations: When a Join Proxy supports one variation for a particular IP version and transport (TCP, UDP stateful/stateless), than it can support all current and future variations for the same IP version and transport without the need for Join Proxy software or configuration updates.

To support agnostic implementations, variations can only differ in the payload of messages carried across those TCP/UDP connections, but not the transport mechanisms used. New transport mechanisms can not be variations, but need to be so-called contexts.

The choice of encoding of variations into different discovery methods also needs to ensure that it can be discovered by legacy Join Proxy implementations.

Initial support for variations in proxies does create additional coding effort: When a pledge or Registrar-Agent connects to a Join Proxy with the need to use a specific variation on a registrar, then the Join Proxy needs to understand which variation that is, so that it can connect the pledge or registrar to a registrar supporting that variation. This requires that proxies create per-variation responder sockets.

2.2. Functional Summary

This document specifies a set of IANA registry tables for BRSKI. These tables allow to define the attributes for different registry mechanisms to announce and discover different BRSKI role responders as well as their variations. Defining these via registry tables maximizes consistency across discovery mechanisms and eases support of variations across different discovery mechanisms.

Using the discovery information specified through these tables, this document specifies details of selection and fail-over when discovering more than one interoperable and available responder, These procedures intend to provide resilience and scalability of BRSKI services not possible without dynamic discovery mechanisms.

Finally, this document specifies procedures for Join Proxies to discover variations of registrars using any discovery mechanism, annnounce them to pledges - and connect a pledge accordingly to the right registrar based on the variation required by the pledge. These procedures allow to introduce new variations of BRSKI without need to upgrade proxies.

3. Specification

3.1. Data Model

BRSKI Discovery is about discovery of one or more instances of responders supporting a specific BRSKI role - and determining whether that responders variation of BRSKI protocol options is compatible with / desired by the connection initiator. This section gives the conceptual overview of how this is achieved.

3.1.1. Roles and Services

In this document, a service is a specific functionality provided by a responder over a network socket using a particular transport/security/session stack (such as TCP, UDP, COAP, DTLS).

In this document, a role is functionality performed by a BRSKI entity either as an initiator or responder. [BRSKI] defines the roles of pledge, Join Proxy, registrar, MASA. [BRSKI-PRM] adds the role Registrar-Agent. Trust anchor is a dependent role required by BRSKI, provided through [EST] or other protocols in [BRSKI-AE].

A single entity may implement multiple roles such as registrars that also often implement the Join Proxy Role or pledges which change stop being a pledge after enrolment but often then become Join Proxies. Future BRSKI documents may introduce additional roles and many of the definitions in this document should be extensible to also support such additional roles.

In this document a service is functionality performed as a result of a network connection from an initator to a responder. The service is commonly named after the role name of the responder, such as Join Proxy, registrar or MASA.

3.1.2. Service Names

The role that a responder socket supports is indicated in each discovery mechanism through an appropriate signalling element. [DNS-SD] calls this signalling element the Service Name. Due to the absence of another equally widely used term for this type of signalling element across arbitrary discovery mechanisms, this document also refers to the role signaling element as the service name, independent of the discovery mechanism. IP Address, IP transport protocol and IP transport protocol port are not part of the Service name and signalled across discovery mechanisms specific signaling elements.

3.1.3. Variations

Variations in the BRSKI protocol such as the choice of encoding of messages or features could impact interoperability between initiator and responder. Initiators need be able to discover and select responders based not only on the desired role, but also based on the best variation for the initiator.

Variations of a role could be indicated by using a different Service Name for every variation, but that approach would have two challenges

1. Service Names in different discovery mechanisms are typically not hierarchical (e.g.: not "role.variation"). Relying only on Service Names would thus require the registration for every variation as a separate Service Name in a "flat" name space; and register them once for each discovery mechanism. In addition, not all discovery mechanism registry rules may look favorably at the registration of Service Names for such protocol variations.

2. Whenever a new variation is introduced, all deployed BRSKI proxies would need to be configured to also proxy this new variation - because new Service Names for the same BRSKI role can be auto discovered by proxies (without additional protocol mechanisms that would be more complex than the variations approach). Most Join Proxies should be able to operate without configuration though.

For these reasons, this document introduces the encoding of BRSKI (role) variations through a secondary signaling element in each discovery method, enabling proxies to transparently support any variation of BRSKI role connections for which they supports proxying.

In addition, variations only need to be registered once in a BRSKI specific registry table introduced by this document, and not once for each current or future discovery method.

A variation is hence specified as describing a combination of signaling choices that a BRSKI connection may use and that impacts interoperability between initiator and responder at the message exchange and encoding level.

3.1.4. Variation Types

Today, BRSKI connections can exchange vouchers in one out of multiple different encoding formats. Independent of that option, the BRSKI connection may also use different commands (so called "Endpoints"). Today, these are based on whether [BRSKI-PRM] is used or not. Finally, and also independent of those two options, the BRSKI connection may use one out of multiple different enrollment protocol options.

This document calls these options "Variation Type", and the above three variation types are called "vformat" for the voucher format, "mode" for the Endpoints being used (such as PRM or not), and "enroll" for the enrollment protocol used.

3.1.5. Variation Type Choices

The actual choices for each of these variation types are hence called "Variation Type Choices": "prm" or "rrm" for the variation type "mode". "cmsj", "cose" or "jose" for the variation type "vformat". "est", "cmp" or "scep" for the variation type "enroll".

"scep" is an example for the ability of the registration to reserve values: it is not adopted by any current BRSKI specification.

3.1.6. Variation Strings

The string name of a variation is as a string concatenating a single variation type choice for every (necessary) variation type. For example "rrm-cmsj-est" could be describing the protocol options used by a [BRSKI] connection pledge to registrar - potentially through a Join Proxy. This string representation of a variation is called the variation string and it is consistently used for signalling across any discovery mechanisms.

When in the future, additional variation types and choices are introduced, existing variation strings must not be changed to allow full backward compatibility with existing/deployed implementations.

For example, when using BRSKI over UDP, today only COAPS is supported, but BRSKI UDP sockets could equally work with QUIC (which runs on top of UDP). At that time, a new variation type of e.g.: "proto" could be introduced with variation type choices "coaps" and "quic". For backward compatibility, "coaps" then needs to be defined to be the default for BRSKI over UDP, which means that existing variation strings such as "rrm-cmsj-est" imply the use of "coaps", whereas the use of QUIC would have to be indicated explicitly via "rrm-cmsj-est-quic".

For variation strings to be semantically unambiguous, the variation type choices across all variation types have distinct names, and the order in which variation type choices are concatenated is the order in which variation types are defined in the according registry table. Hence new variation type choices have to be tail added to the registry table.

Just because a variation name is composed from variation type choices does not mean that an unspecified variation of (random) variation type choices can work without new implementation or specification. Or even make sense. This may be the case, or it may not. This is also the reason why this document specifies a registry that explicitly enumerates all variations that are known to have sufficient specification and will work.

For example, [BRSKI-PRM] is indicated through the variation type value "prm", but it may also requires enhancements to the enrolment protocol used, which is specified in the variation type "enroll", such as new endpoints in that protocol. The required functional semantic implied by the "enroll" variation type value in variations with "prm" is thus a different one than in variations not using "prm". And [BRSKI-PRM] does not necessarily sufficiently specify these enhancements for enrollment protocols that may not have been known or specified by the time [BRSKI-PRM] was written.

3.1.7. Contexts

Variation strings are defined separately for every group of services for which the set of variation strings is or could be different or could have different semantics. A group of services for which the same variation strings are defined is called a Context.

Different list of variation strings are necessary when services have different variation types, different variation type values, different deployed variations or different defaults for the same variation type values and hence different variation strings.

"tBRSKI" is the context covering [BRSKI] connections (which are using TCP, hence the "t") from pledge to Join Proxy or registrar and from Join Proxy to registrar connections.

"cBRSKI" ("c"onstrained BRSKI) is the context covering [cBRSKI] connections (using UDP) from pledge to Join Proxy or registrar and from Join Proxy to registrar connections.

"BRSKI-PLEDGE" is the context covering pledges using [BRSKI-PRM] for connections from Registrar-Agents to pledges. It can equally cover in the future through variations the discovery of [BRSKI] pledges for connections to them for other purposes - by introduction of appropriate variation types and values for such additional purposes.

This document does not define variations for different end-to-end encryption mechanisms. However, the mechanisms described here can also be used to introduce backward incompatible new secure transport options.

Also, this document does not introduce contexts for discovery of other BRSKI roles beyond those mentioned, such as discovery discovery of MASA by registrars. However, the registries introduced by this document are defined such that those can be introduced later as well through additional registry entries and specifications.

3.1.8. Discussion

Variations as defined by this document only cover protocol options that proxies can transparently support so that the definition of variations allows to make proxies automatically extensible.

Other responder selection criteria such as different responder priority or performance based selection (called weight in [DNS-SD]) are not covered by the variation concept but can be used without change in conjunction with variations. Some selection criteria may also only work with discovery mechanisms that rely on specific procedures. Network distance to responder can for example only be well supported by discovery mechanisms that can support per-hop forwarding between initiator and responder, such as [GRASP]. Any of these criteria will work unchanged with the introduction of Variations. Variations are simply one more selection criteria.

Differences in the supported transport stack of a responder are typically included as a signaling element of the discovery method: Whether TCP or UDP or another IP transport protocol is used, and whether the responder uses IP or even another network layer protocol.

In "sane" services where a change in transport spec does not imply a change in signalled messages and their semantics, gateways could transparently proxy from IP and vice versa or even between TCP and some other IP transport protocols such as SCTP. However, this is out of scope of this specification.

The procedures specified in [cPROXY] would allow not only to run a transport stack of COAP over DTLS, but equally any other transport stack over UDP, such as QUIC - without any changes to the Join Proxy implementation or configuration when following the procedures described in this document. All that is needed would be to introduce appropriate registration entries for the registry tables specified in this document (e.g.: add new Variation Type for transport and choices such as "coaps" or "quic").

3.1.9. IANA Registry Tables

This sub-section re-states and summarizes the desired Data Model introduced before in it’s instantiation of the IANA Registry (and sub-registries) requested by this document, the so-called "BRSKI Discovery Parameters" registry (Section 5.4).

The detailled requirements for registrations and initial content of the registry tables are defined in the IANA section for it.

3.1.9.1. Discovery Mechanisms registry

The main Discovery Parameters table ({#reg-discovery}) simply defines the abbreviations for the Discovery Mechanisms specified for use with BRSKI Discovery: CORE-LF, DNS-SD and GRASP. It allows to add new Discovery mechanisms and track where they are specified - including outside the IETF.

This table is called the "main" table because the "BRSKI Discovery Parameters" is a registry with sub-registries and one table has to be associated with the registry itself instead of a sub-registry.

3.1.9.2. Contexts sub-registry

The IANA "Contexts" sub-registry, Section 5.4.1 registers names for different groups of services in the overall BRSKI framework: BRSKI for [BRSKI] Registrar->Proxy->Pledge, cBRSKI for constrained BRSKI ([cBRSKI] / [cPROXY]) Registrar->(Proxy)->Pledge, and BRSKI-PLEDGE for [BRSKI-PRM] discovery of pledges.

Grouping services allows to limit registrations/definitions in other registry tables to these Contexts as opposed to repeating them for every service (Registrar, Statelss-Registrar, Proxy for example).

Distinguishing BRSKI from cBRSKI is also necessary so that the different Transport Stack parameter(s) between the two can be appropriately included into the registries (see Section 5.4.2)

The Contexts sub-registry also registers the list of Variation Types defined for each context. The Variation Types are the different aspects of the BRSKI protocols for which different choices have been specified or could be specified.

3.1.9.3. Service Names sub-registry

The IANA "Service Names" sub-registry, Section 5.4.2 registers the "Service Names" used to discovery the different BRSKI services for each context and Discovery Mechanism. It also specifies the necessary (Transport Stack) Parameters that needs to be used in the discovery mechanism to indicate (discover or recognize in discovery replies) the service. In the initial table as specified by this document this is primarily used to distinguish BRSKI services which use TCP and cBRSKI services which use coaps, also represented as UDP in some discovery mechanisms (such as DNS-SD).

3.1.9.4. Variation Type Choices sub-registry

The "IANA Variation Type Choices", Section 5.4.3 depends on the definition of Variation Types from the Contexts sub-registry, Section 5.4.1 by defining the specified choices for each Variation Type and registers the specification(s) where they are defined.

Registration of a Variation Type Choice is per Context, so that different Contexts can have different specification (and hence procedures) for the same Variation Type. In the initial registry content requested by this document, this is used to refer to the separate specification for BRSKI variations and cBRSKI variations.

3.1.9.5. Variations sub-registry

The IANA "Variations" sub-registry, Section 5.4.4 registers the so-called "Variation String" encoding of variations of BRSKI protocols services for each Context. This is ultimately the encoding specified in this document to be used to signal across all Discovery Methods the actual Variations supported by a Service Instance. This sub-registry depends on the definitions of the prior registry tables, which is why it is last.

A Variation is a combination of one Variation Type Choice for every Variation Type. Effectively, it describes one way that a particular BRSKI service can operate. With the Variation Types "mode", "vformat" (voucher format) and "enroll" (enrollment protocol) specified in this document, it is possible to specify a Variation that has a different overall procedure: "mode" can be "rrm" to indicate the pledge driven progression of BRSKI (as in [BRSKI] and [cBRSKI]) or the "prm" Variation Type Choice as specified in [BRSKI-PRM]. "vformat" specifies one one the encoding options possible for the vocher exchanged in BRSKI, and "enroll" specifies the enrollment protocol, which logically is separate from BRSKI but factually included into BRSKI because it so far not only shares most of the time the secure Pledge to Registrar connection, but it also is an interoperability requirement between Registrar and Pledge: Both need to support the same encoding for a BRSKI exchange to finish successfully.

Some Variation Type Choices may actually impact other Variation Types, for example, "prm" does impact also details relating to the enrollment protocol, and [BRSKI-PRM] accordingly speficies how the [EST] ("enroll" = "est") needs to be modified to operate "prm". For this reason, the Variations sub-registry only enlists Variations that are explicitly specified somewhere as actually being a working combination of Variation Type Choices. Instead of simply permitting implementations to announce all the Variation Type Choices seprately - and hope any combinations actually do work.

Variations are encoded as Variation Strings which are combined by concatenating the Variation Type Choice strings with "-", leaving out those Variation Type Choices that are the "Default" (Dflt in the sub-registry). This encoding scheme allows to maintain unchanged old Variation Strings when introducing new Variation Types in a backward compatible fashion. The backward compatible choice for a new Variation Type is marked as "Dflt", does hence not need to be included in the Variation Type Choice, and hence th pre-existing Variation Strings without the new Variation Type can continue to exist and be used without changes.

3.2. Redundant Discovery and Selection

The following subsections describe requirements for resilient and scalable responder selection. Resilience is supported by automatically selecting the currently best available responder. Scalability of simultaneous sessions is supported through distributing the connections from multiple initiators to different responders if so desired through operator configuration of the discovery methods parameters.

At the time of this specification, the relevant initiators are BRSKI pledges, Join Proxies and Registrar-Agents, the relevant responders are Join Proxies and BRSKI Registrars. Nevertheless, the rules defined in this document can equally apply to other BRSKI connections if and when discoverable and redundant services are desired and added to the registries created by this document. For example discovery of MASA by BRSKI Registrars.

Note that this specification does not mandate support for specific discovery methods in BRSKI implementations, because this is specific to the target deployment scenarios - hence the option to support different discovery methods.

Note that while pledges are discoverable in the context of this documents technologies, this section and its subsections do not apply to discovery of pledges because there is no redundancy involved, and selection of pledges is also only by their ID and not by their supported variation.

3.2.1. Responder Selection

If more than one responder is discovered by an initiator, then the initiator SHOULD support to sequentially attempt to connect to each feasible responder exactly once until it successfully connects to one. If it fails to connect to any feasible responder, the initiator SHOULD wait until at least 30 seconds have elapsed since the start of the last round and update its discoverable responder information from the discovery mechanism if that is not already happening automatically by the chosen discovery method before it restarts connection attempts.

A responder is feasible if it supports one or more of the variations requested by the inititor.

The order of responders to attempt connections to is derived from two criteria: preference and weight.

Preference order is foremost determined by the responders preference across the variations it supports. Within the set of of responders with the same preference by the initiator because of their variation, the preference is further determined from discovery method specific preference parameters such as the "priority" parameter in DNS-SD, or possible future distance parameters in discovery mechanisms like GRASP.

If a responder socket offers more than one variation supported by the initiator its preference order is calculated from the most preferred variation supported by it.

Within a set of two or more responders with the same preference, the initiator MUST pick at random, especially after power-on or other reboot events. This is to ensure that those events have the chance to overcome possible persistent problems when persistently choosing the same first responder. If deployments desire reproducable and predictable ordering of connection attempts by initiators then they have to use the discovery specific mechanisms, such as a different priority" parameter for each responder in DNS-SD to create such a strict ordering across the different responder.

Initiators SHOULD support to take discovery mechanism specific weighting into account when determining the order of responders with the same preference, such as the "weight" parameter in DNS-SD.

Support for the so far described resilient selection of responders SHOULD support selection amongst at least 4 and no more than 10 responders with one or more supported variation for each supported IP address family (v4 and/or v6). If more responders are discovered for the preferred variation(s) of the initiator, then it SHOULD pick a random subset of those responder announcements to select from.

4 Responders for a specific variation are a typical minimum resilience setup in a larger network setup, in which 2 responders serve as redundancy at the responder host level and the other 2 responders provide redundancy against network connectivity failure to those first two responders. Intra-DC and Inter-DC service redundany is a simple example of such a setup.

3.2.2. Service Announcements

Responder selection as described in Section 3.2.1 needs to deal with unresponsive responders because service announcements may be stale. This happens when service announcements only loosely track aliveness of a service process.

In typical implementations, service announcement may be activated when the service process starts, and stopped, when it stops. Problems such as a hanging (unresponsive) service process will not be reflected in the service announcement setup. In addition, caching of service announcements, such as through the DNS TTL field are a further possible cause of assuming service aliveness that is not correct. Only actual connection probing or other similar tracking can determine if a service responder is responsive to the level of accepting connections.

Responders intended to be used in resilient deployments SHOULD therefore ensure that their service announcements are not active when the responder died or would have failed to successfully accept connection for 120 seconds or more. This can be implemented for example by connection probing once every 30 seconds and withdrawing the service announcements when this fails or by other forms of tracking responsiveness of the responder functionality.

The better service announcements indicate actual aliveness of the service instances, the faster service selection will be. In addition, in large networks, backup/standby service instances can then be implemented by tracking primary service announcemements and activating the backup only when the primary ones fail. Such dynamic backup can further reduce the overall load on the discovery mechanism system used and on initiators.

3.2.3. Responder Selection in Proxies

Unless amended by the requirements listed below, proxies SHOULD follow all the descripton from Section 3.2.1. Note that the randomn selection of responders with the same preference also applies to stateful proxies and ensures load balancing (including weighting) across multiple simultaneously connecting pledges.

Stateful proxies SHOULD optimize selection of responders for each variation across connections for multiple pledges instead of starting the sequence of responders to try from the highest precedence anew for every new connecting pledge - and repeatedly run into timeouts for each new connecting pledge when those primary responders time out on connection attempts because they are unresponsive or unreachable. Instead, after a responder first fails to connect, the Join Proxy SHOULD skip this responder in further connection attempts for other connecting pledges.

Stateless proxies cannot learn unresponsiveness or unreachability of a responder through connection attempts. Instead, they SHOULD perform a stateless responsiveness/reachability check for each responder that the Join Proxy is actively forwarding packets to from one or more pledges. If no packets are returned from such a responder over a period of more than 30 seconds, then the responder SHOULD be considered unreachable for at least 180 seconds. Unreachability signaling received in response to packets sent to the responder SHOULD trigger this unreachability status after it persists for 10 seconds.

Load balancing as described in Section 3.2.1 is NOT RECOMMENDED for stateless proxies because per-pledge stateless load balancing may involve more processing complexity than feasible for proxies on constrained devices. To avoid changing the selection of active responders when one responder becomes unresponsive, a "stable hash" approach would have to be used, such as described in [HRW98], which is used for example by [I-D.ietf-bess-evpn-fast-df-recovery]. Supporting weights with stateless proxying is even more complex. Instead of load balancing, responders simply need to be designed to scale to the maximum amount of simultaneous initiator connections necessary when supporting stateless proxying mode.

3.2.4. Protection against malicious service announcements

Initiators including proxies SHOULD always pick the highest possible priority and weight parameters possible in the discovery mechanism used that allows to support the desired preference goals. For example, any primary initiator should select the highest numerical values possible.

This recommendation is intended as a protection against erroneous, but not malicious service announcements whenever the default priorities are lower than the maximum priority. It can also serve as a weak protection against malicious announcements because with the selection rules required by this document, initiators still have the highest chance of picking the non-malicious announcement.

While being weak, this recommendation can still be better than nothing against such malicious announcement. These recommendations SHOULD be superseded by better recommendations for more narrowly scoped deployment scenarios.

3.3. Join Proxies Support for Discovery and Variations

3.3.1. Join Proxy support for Variations

A Join Proxy compliant with this specification MUST support announcing its Join Proxy responder socket(s) to which pledges can connect via at least one of the discovery methods included in the registry tables specified in this document. The Join Proxy MUST announce the variation(s) supported on its responder socket(s) according to the registry table entries.

A Join Proxy SHOULD support to pass packets for any variation for which it has discovered one or more registrar sockets supporting that variation without the need for the variation to be known at the time of implementation of the Join Proxy or configured on the Join Proxy. If a Join Proxy supports this requirements, this is called support for "arbitrary variations". Supporting this requirement requires the Join Proxy to discover registrar(s) and their supported variation(s) via one or more of the discovery mechanisms included in the registry tables specified in this document.

Arbitrary variations support allows to deploy proxies once and add pledges and registrars supporting new variations later - without upgrade or change of configuration of proxies.

3.3.2. Registrar Operations Modes

Proxies may use different approaches to connect to registrars. The following subsections discuss the primary relevant options.

3.3.2.1. Direct Connection Mode

In one Join Proxy implementation option called "direct connections", the Join Proxy creates for every discovered registrar socket a separate Join Proxy responder socket. It announces this socket with the same set of parameters as it did learn from the registrar's service announcement, except for the appropriate Join Proxy service name and socket parameters (IP address, port number). When a pledge connects to that socket, the Join Proxy passes traffic for that pledge's connection to and from the respective registrar socket.

When using the direct connections approach, the task of selecting the best registrar socket for a particular variation is left to pledges because they are exposed to at least the same number of service announcements from proxies as proxies see service announcements from registrars - and the pledge has to select the best available one from them.

To reduce the number of sockets that have to be announced by proxies when using direct connections and to also reduce the number of responder sockets that need to be maintained by a Join Proxy operating in this approach, these proxies SHOULD limit the number of registrar sockets they maps to between 4 and 10 best registrar sockets as described in Section 3.2.1 per variation.

3.3.2.2. Best Registrar Selection Mode

In the implementation mode "best registrar selection", the Join Proxy creates a separate responder socket for a set of all registrar sockets that it discovers and that announce support for the same set of variations.

For pledges to discover the Join Proxy, the Join Proxy then announces this socket with the parameters of the best discovered registrar socket, replacing the service name and network parameter names with those for its Join Proxy responder socket as in the case of a direct connection.

The Join Proxy then connects pledges to the best available registrar socket from that set when it receives a connection to that socket.

When performing best registrar selection for that connection, the Join Proxy has to perform selection of the best availalable responder as described in Section 3.2.1.

Using newly discovered responders in stateless proxies supporting best registrar selection must be done carefully. Consider the common case that service announcements for a primary responder did stop due to network issues, now the Join Proxy starts to send packets to a secondary responder, and then the primary responder becomes reachable and the Join Proxy sees service announcements for it. If the Join Proxy would now start to forward packets from pledges to this primary responder due to its higher precedence, then this could unnecessarily break ongoing connections from clients whose packets are currently forwarded to the lower preference Join Proxy. Direct connection mode does not incur this problem, because the pledge can select another Join Proxy responder socket when it discovers the first one to be unresponsive or erroneous.

Replacing a selected responder in a stateless Join Proxy with a better one SHOULD hence only be done if no packets have been exchanged between pleges and the current selected responder through the Join Proxy for more than 300 seconds. This long timeout specifically intends to not break connections in which the registrar keeps the pledge waiting for an administrative response such as an operator performing a policy validation for enrollment.

In addition, stateful proxies implementing best registrar selection SHOULD optimize selection of registrar for each Join Proxy responder socket across connections for multiple pledges instead of starting the sequence of responders to try anew from the highest precedence registrar for every new connecting pledge - and repeatedly run into timeouts when one or more of the best registrar time out on connection attempts because they are unresponsive or unreachable. Instead, after a responder first fails to connect, the Join Proxy SHOULD skip this responder in further connection attempts for other connecting pledges and re-consider it only for new connection attempts after at least 60 seconds.

3.3.2.3. Proxy in Service Name Only Mode on Registrars

Registrars that implement support for connections from stateful proxies and/or from pledges may minimize their Join Proxy implementation work by only implementing the appropriate service name announcements for the same socket to support connections from both: announcements as a registrar for connections from proxies and announcements as a Join Proxy for connections from pledges. No additional sockets or other Join Proxy specific packet processing code is required to support this.

Registrars that implement support for connections from stateless proxies can share that implementation for connections from pledges by also implementing simple UDP<->JPY header conversion (see [cPROXY]). Nevertheless, they do need to do this via a separate socket and hence need to announce the two sockets separately: UDP for connections from pledges with the Join Proxy service name, and UDP with JPY header for connections from stateless proxies with the stateless registrar service name.

Proxy functionality that is implemented as described here on registrars is called "proxy in service name only mode", because such an implementation cannot discover, select and fail over between different registrars. Such proxies in name only therefore do not share requirements against discovering and selecting registrars described for the prior specified modes.

Like other proxies, proxies in name only SHOULD nevertheless track aliveness of their registrar function and withdraw its service announcements (both as Join Proxy as well as as registrar) when it does not run, fails or becomes unresponsive.

Proxies in name only SHOULD default to the same discovery method priority and weight parameter as those configured for the registrar service announcements. This is so that in the absence of separate proxies in the network selection of registrars co-located proxies would follow the same criteria as those used by proxies and which use the registrar service announcement parameters.

3.3.2.4. Proxy Mode Discussion

This document defines no requirements against the implementation mode for proxies. Those are left for solution or deployment (profile) specifications. Instead, this section discusses some considerations for those choices.

The list of possible modes presented is exemplary and not meant to be exhaustive. Other modes are equally able to support the requirements, such as mixtures of the described modes. Likewise, introduction of new variations may not only work well via arbitrary variation support in proxies, but through explicit configuration of variations on proxies - this all depends on the target deployment environment. The presented modes where choosen primarily as the ones providing most configuration free deployment options and for registrar implementations most simplicity in implementation.

If a deployment has a larger number of service announcements and (extremely) constrained pledges, direct connection mode may be inappropriate because it shifts the burden of best available

discovery and selection and onto the pledge. If simultaneously proxies in such deployments can support more scalable complex implementations, then best registrar selection mode may be most appropriate.

In environments, where all pledges are expected to become proxies after enrollment, implementers may simply want to implement the option for which both pledge and Join Proxy code together is easiest to implement.

Even on registrars, proxy in service name only mode may not suffice deployment requirements or provide best redundancy. For example, the co-located registrar may incur problems, not applicable to alternative registrars, such as for example Internet connectivity problems to MASAs when different registrars have different Internet connectivity. If the registrar co-located Join Proxy is then still the only Join Proxy available to some candidate pledges, then this Join Proxy needs to be able to connect to an alternative registrar, which would not be possible if it was a proxy in service name only.

Likewise, proxy in service name only mode will disturb the introduction of new variations on pledges and other registrars in the network if the registrar node implementing proxy in service name only mode becomes a necessary Join Proxy for a pledge requiring a variation not supported by this registrar, but by another registrar that would only be reachable through this registrar node. Therefore, Join Proxy in name only mode is best suited for node types not deployed on an edge of the network where a future variety of pledges may connect to, and those pledges will require the use of a Join Proxy.

3.3.3. Extensibility to non BRSKI services

Join Proxy implementations using the procedures described in this document can easily be reused for any other protocols beside BRSKI as long as they use TCP or UDP. For this, it would simply be required that the Join Proxy can be configured with pairs of service names other than those used by tBRSKI/cBRSKI: A service name to discover, and a service name to use for the Join Proxy responder socket service announcements.

3.3.4. Scaling service discovery and selection

Discovering and selecting an available service instance can become a design challenge in large networks with many redundant service announcements.

Consider for example the common case of allowing BRSKI registration in a network with many geographically spread out sites such as in enterprise, industrial or building construction environments. During initial bringup of such sites, Internet connectivity may be non-existing, or intermittant, and hence one or more local registrar in each such site is higly desirable. Such registrars may of course require private mobile network connectivity to MASA, or rely on out-of-band provisioning of vouchers.

Later, when such a site does get a well working wide-area network connection, it may be more appropriate to use more centralized registrars, but a local registrar as a backup may be considered useful. However, if the service announcements of such per-site decentralized registrars would be discoverable across the whole geographically spread out network, then this could introduce a potentially significant overhead to the service announce and discovery system when for example more than 100 registrar service announcments exist in the network, and pledges/proxies connect to them.

Such large number of redundant service announcements is typically highly undesirable, and appropriate configurations of service discovery mechanisms need to be used to avoid them. For example, in GRASP, service announcements can be scoped to small hop counts, Anycast addressing can be used to make all decentralized registrars overload the same ip address, and hence make them all share the same service announcement.

3.4. Discoverable BRSKI Pledges

3.4.1. BRSKI-PLEDGE context

BRSKI-PLEDGE is the context for discovery of pledges by nodes such as Registrar-Agents. Pledges supporting [BRSKI-PRM] MUST support it. It may also be used by other variations of BRSKI outside of the PRM use case, for example for inventorizing pledges.

Pledges supporting BRSKI-PLEDGE MUST support DNS-SD for discovery via mDNS, using link-local scope. For DNS-SD discovery beyond link-local scope, pledges MAY support DNS-SD via [DNSSD-SRP], using automatic discovery of the SRP server and registering the below defined DNS-SD data with it.

These DNS-SD requirements are defaults. Specifications for specific deployment contexts such as specific type of radio mesh network solutions may need to specify their own requirements overriding or amending these requirements.

Pledges MUST support to be discoverable via their DNS-SD service instance name.

Pledges SHOULD support to be discoverable via DNS-SD browsing, so that Registrar-Agents can find unexpected pledges or can enumerate expected pleges more easily, especially in the presence of multiple different subnets and use of mDNS. A pledge can also only be found by browsing if it is not possible for the owner to aquire serial-number information of a pledge by the time BRSKI-PRM needs it (to create a service instance name).

When pledges are discoverable vis DNS-SD browsing, the "brski-registrar" PTR service name is a so-called shared resource record. When it is requested via mDNS (multicast), all pledges supporting BRSKI-PLEDGE and browsing will respond simultaneously, potentially creating congestion/contention. To avoid this, [mDNS] specifies the following requirement: "each responder SHOULD delay its response by a random amount of time selected with uniform random distribution in the range 20-120 ms."

It is equally RECOMMENDED to apply the same random delay rule for answers to multicasted or flooded queries in other discovery mechanisms that have the same response burst problem - even when they do not specify such a mechanism, such as in GRASP.

If browsing is not desired by a pledge, the pledge does simply not respond to queries for the "brski-registrar" service name in mDNS or other discovery mechanism queries for the equivalent service name, or does not register its PTR RR for this service name when using unicast DNS-SD via [DNSSD-SRP]. This does not affect operations for the service instance name.

This specification does not introduce the procedures to discover pledges via CORE-LF or GRASP because it is unclear if there is currently demand for this, and because it can easily be added via additional specs and adding entries to the registry. For CORE-LF, browsing of entries can be supported via CORE-RD ({RFC9176}}), with appropriate auto-discovery of the CORE-RD server. For GRASP, this could be done via a method mapping DNS-SD into GRASP, such as [I-D.eckert-anima-grasp-dnssd], specifically to support browsing of pledge instance names.

3.4.2. Service Instance Name

The service instance name chosen by a BRSKI pledge MUST be composed from information which is

* Easily known by BRSKI operations, such as the operational personnel or software automation, specifically sales integration backend software.

* Available to the pledge software itself, for example by being encoded in some attribute of the IDevID.

Typically, a customer will know the serial number of a product from sales information, or even from bar-code/QR-codes on the product itself. If this serial number is used as the service instance name to discover a pledge from a Registrar-Agent, then this may potentially (but unlikely) lead to (duplicate) replies from two or more pledges having the same serial number, such as in the following cases:

1. A manufacturer has different product lines and re-uses serial-numbers across them.

2. Two different manufacturers re-use the same serial-number space.

If pledges enable browsing of their service instance name, they MAY support DNS-SD specified procedures to create unique service instance names when they discover such clashes, by appending a space and serial number, starting with 2 to the service instance name: "<service-instance-name> (2)", as described in [DNS-SD] Appendix D.

Nevertheless, this approach to resolving conflicts is not desirable:

* If browsing of DNS-SD service instance name is not supported, Registrar-Agents would have to always (and mostly wrongly) guess that there is a clash and (mostly unnecessarily) search for "<service-instance-name> (2)".

* If a clash exists between pledges from the same manufacturer, and even if the Registrar-Agent then attempts to start enrolling all pledges with the same clashing service instance name, it may not have enough information to distinguish pledges other than by the randomn numbering. This would happen especially if the IDevID from both devices (of different product type), had the same serial number, and the trust anchor certificate of both was the same (e.g. same root CA certificate), which is likely when they are from the same manufacturer. Even if some other IDevID field was used to distinguish their device model, the Registrar-Agent would not be able to determine that difference without additional vendor specific programming.

In result:

* Vendors MUST document a scheme how their pledges form a service instance name from information available to the customer of the pledge.

* These service instance names MUST be unique across all IDevID of the manufacturer that share the same trust anchor.

The following mechanisms are recommended:

* Pledges SHOULD encode manufacturer unique product instance information in their subject name serialNumber. [RFC5280] calls this the X520SerialNumber.

* Pledges SHOULD make this serialNumber information consistent with easily accessible product instance information when in physical possession of the pledge, such as product type code and serial number on bar-code/QR-code to enable [BRSKI-PRM] discovery without additional backend sales integration. Note that discovery alone does not allow for enrollment (so it does not introduce a security risk by itself)!

* Pledges SHOULD construct their service instance name by concatenating their X520SerialNumber with a domain name that is used by the manufacturer and thus allows to disambiguate devices from different manufacturer using the same serialNumber scheme, and hence the likelihood of service instance name clashes if manufacturer names are not used.

* Pledges MAY re-use the service instance name as their host name in their AAAA or A RRs.

3.4.3. Example

This section discusses an example manufacturers approach using the recommendations from above. Figure 1 shows the different data involved in DNS-SD records for a pledge from manufacturer "Example".

```
1. Manufacturer published information:

  1.1 IDevID trust anchor certificate.

  1.2 IDevID X520 identification schema:
    Brand: Example
      O = example.com, serialNumber = "PID:Model-<PID> SN:<SN>"
      ; Explanation:
      ; SN = Serial Number, PID = Product Identifier
      ; O = Organization

  1.3 DNS-SD Instance Schema:
    <X520SerialNumer>.example.com

2. Example Purchase Order Pledge Information
   Brand: Example, Model: 0815, Serial: WLDPC2117A99

3. DNS-SD Instance string:
  "PID:Model-0815 SN:WLDPC2117A99.example.com"

4. DNS-SD RR for the pledge (using mDNS, hand hence .local):
  ; PTR RR to support discovering the pledge through browsing,
  ; e.g. when the serial number is not known in advance
  _brski-pledge._tcp.local  IN PTR
    PID:Model-0815\\ SN:WLDPC2117A99\\.example\\.com._brski-pledge._tcp.local

  ; SRC and TXT RR for the service instance name
  PID:Model-0815\\ SN:WLDPC2117A99\\.example\\.com._brski-pledge._tcp.local
    IN SRV 1 1
    PID:Model-0815\\ SN:WLDPC2117A99\\.example\\.com.local
  PID:Model-0815\\ SN:WLDPC2117A99\\.example\\.com._brski-pledge._tcp.local
    IN TXT ""

  ; AAAA address resolution for the target host name
  PID:Model-0815\\ SN:WLDPC2117A99\\.example\\.com.local
    IN AAAA fda3:79a6:f6ee:0000::0200:0000:6400:00a1

5. Example Pledge IDevID certificate information:
    ; Format as shown by e.g.: openssh
    Subject: serialNumber = "PID:Model-0815 SN:WLDPC2117A99",
     O = example.com, CN = Model-0815
```

Figure 1: Example IDevID and DNS-SD data

Using the information from the above Example picture, a Registrar-Agent implementation operates as follows.

1. Initially, it is configured with the Manufacturer published information (1.), and as not shown it will likely have a list of such information for various brands of products.

2. After a pledge purchase and initial physical deployment, it is provided with the Purchase Order information for the pledge (2.), this order information tells it, that the Manufacturers Brand is "Example", that the pledge product Model <PID> is "0815" and its Serial Number <SN> "WLDPC2117A99".

3. It looks up the IDevID X520 identification schema for brand "Example" and uses the template value for the serialNumber field together with the pledge information values from (2.) for O, <PID> and <SN> to determine the DNS-SD Instance of the pledge (3.). That instance is the concatenation of the X520 serialNumber value of the pledge with the Organization domain name of the manufacturer. With the Organization being a global DNS domain "example.com", including this in the Instance makes it unique across manufacturers.

4. It then uses standard DNS-SD via mDNS to find the pledge, using the DNS-SD Resource Records (RR) shown in 4.

5. Once it receives a response from the device claiming to be the pledge, it can use any appropriate authentication mechanism to validate ownership of the IDevID private key. In [BRSKI-PRM] this is done through the initial TLS handshake in which it learns the presumed IDevID of the pledge. When the signature in the IDevID matches the public key of the desired Manufacturer from (1.1), then it trusts that this pledge is from that manufacturer. When the O and serialNumber of the certificate match what it constructed from the <PID>/<SN> values from purchase information from (3.) then it also trusts that this is indeed the pledge it was looking for.

Instead of using sales receipt information, the customer may also use scanned QR/Barcode or RF-Tag information from packaging after receiving shipments for example for step 2. Scanning packaging information will likely will introduce additional complexity because the manufacturer name can often only be derived from information such as EAN-13 barcodes.

The process steps may also be simpler if the customer can know the IDevID of the pledge through the purchase process instead of having to match the IDevID from sales receipt information (step 2).

The process steps may also become more complex. The manufacturer may have multiple brands using the same trust anchor. Or the manufacturer may have certificate chains and different production sub-companies may use different sub-CA certificates in the signing chain. Or the serialNumber alone is not unique across the certificate chain, but further Subject fields of the certificate are required for a unique identification, such as the O)rganization field. It could contain for example one out of multiple brand names that use simple numerical serialNumber formats and hence could collide. None of such complexities are desirable for new designs, but they may be necessary to support BRSKI for existing products.

For the described mechanism to work, the manufacturer does not necessarily have to own a domain name such as "example.com" in the example. Owning a domain name and using it for the DNS-SD Instance Schema is just a simple way to ensure usage of a unique Instance Schema - if all manufacturers agree to use this approach.

The RR used to discover the pledge by its serial number is the "IN SRV" RR. The "IN PTR" RR is optional and allows for the pledge to be discovered with DNS-SD browsing, which is necessary if the pledges serial number information can not be known by the time the pledge needs to be enrolled by a Registar-Agent.

The instance string is also re-used in the host name of the "IN SRV" RR and hence the domain name of the "IN AAAA" RR. The is just an option, and the pledge can use whatever host name it wants.

3.4.4. WebPKI derived instance schema

There is currently no automated mechanism to avoid the configuration of manufacturer trust anchor certificates in BRSKI components that need to authenticate pledges. However, the configuration of additional instance schemas for different manufacturer device names in BRSKI equipment could be avoided if it is deemed appropriate by vendors and operators of BRSKI-PLEDGE installations to rely on WebPKI trust anchors.

The trust anchor certificate itself (or a sub-CA in the certificate chain) would then have to have a WebPKI trust anchor signature and a DNS Name that can easily be identified as being used for IDevID, such as "*.idevid.example.com". And the implied schema for the instance string is then "<<X520SerialNumer>.DNS-name>", authenticating instance names of the format "<X520SerialNumer>.idevid.example.com>".

Obtaining a WebPKI signature for their trust anchor for these wildcard domain names from a WebPKI trust anchor is the added effort for manufacturer of this scheme.

3.5. Variation signaling and encoding rules for different discovery mechanisms

3.5.1. DNS-SD

3.5.1.1. Signaling

The following definitions apply to any instantiation of DNS-SD including DNS-SD via mDNS as defined in [DNS-SD], but also via unicast DNS, for example by registering the necessary DNS-SD Resource Records (RR) via [DNSSD-SRP].

Because of the different options of how to run DNS-SD, the requirements in this document do not guarantee interoperability when using DNS-SD. One side could use unicast DNS-SD, the other mDNS, and there may be no mapping between the two. Therefore the recommendations in this document need to be amended with deployment specific specifications / requirements as to which signaling variation, such as mDNS or unicast DNS with SRP is to be supported between initiator and responder. When using unicast DNS (with SRP), additional mechanisms are required to learn the IP address(es) of feasible DNS and SRP servers, and deployment may also need agreements for the (default) domain they want to use in unicast DNS. Hence, a mandatory to implement (MTI) profile is not feasible because of the wide range of variations to deploy DNS-SD.

In the absence of overriding deployment profile requirements, implementations are RECOMMENDED to support mDNS and MAY support [DNSSD-SRP] and fall back to mDNS if [DNSSD-SRP] fails to work, e.g.: it fails to discover SRP server and/or default domain.

3.5.1.2. Variation String Encoding

Variation Strings from the IANA registry Table 8 are encoded as DNS-SD Keys with a value of 1 in the DNS-SD service instances TXT RR using the shortened encoding of "key" instead of "key=1". In result, the value of the TXT RR is a sequence of zero terminated strings, each one indicating a single supported variation type choice.

A variation may have the option of being represented by the empty string "". This is not allowed in the DNS-SD encoding, and instead the alternative variation string MUST always be used for DNS-SD.

Variation strings in DNS-SD are case insensitive as required by DNS-SD. It is RECOMMENDED to only announce lowercase variation strings in DNS-SD.

The use of variation strings can easily break the DNS-SD rule that they keys should be no more than 9 characters long. This is justified by the absence of value fields to keep the total length of the TXT RR reasonably short.

3.5.1.3. Service Instance and Host Names

To be able to specify for each responder socket individually its supported variations as well as its selection criteria (priority weight), it needs to be represented in DNS-SD as a service instance name with an SRV and TXT RR. In BRSKI-PLEDGE Section 3.4 the service instance name is significant as it is what a Registrar-Agent may need to discover, but in tBRSKI and cBRSKI it is merely an artefact of DNS-SD encoding: Unlike typical user-centric DNS-SD use-cases, there are no users that need to make sense of the meaning of the service instance name, for example to know, which printer to pick. Only operators may need to look at them for troubleshooting. The choice of instance name (the first component of a service instance name) is hence arbitrary. The same is true for the host names used in the DNS-SD records for BRSKI.

Registrars SHOULD support automatic generation of their service instance name for their DNS-SD operation to avoid additional need for operator configurations. Registrars SHOULD likewise support the configuration of such a name - if the operator so desires to support operational troubleshooting.

If the host on which the registrar is running already has a DNS host name for the IP addresses used by the registrar and for the desired DNS method (mDNS = .local, unicast DNS = default domain), then the registar SHOULD be able to use that host name as the target domain name in the SRV RR. This requirement avoids the unnecessary addition of DNS A/AAAA RRs because of the registrar, when useable RRs already exist.

If such a DNS RR does not exist, but a DNS host name for a different DNS method, or a different set of addresses than used by the registrar, then the registrar MAY be able to use a target domain name derived from that primary domain name by appending a unique name element. This requirement exist to avoid the creation of unnecessarily inconsistent host names.

If no DNS host name exists, the registrar MUST be able to automatically create a DNS host name and the A and/or AAAA RRs for the address(es) used by the registrar for use in the SRV RR target field. This requirement exists to ensure that operators are not unnecessarily required to configure a host name on a system that does not need one - and none is required to run a registrar.

A registrar MAY use any unique identifiers of its host system as its instance name or host name. This can avoid or at least minimize the need to automatically pick another name in case the chosen name is already taken by another system. This for example would happen if a registrar tried to use an instance component such as "registrar" and there is already another registrar. Using a known unique identifier allows a registrar to raise an alert and claim an operational error with a high degree of confidence.

MAC addresses are only unique when an application such as a registrar understands what hardware it is running on, and that the MAC address was assigned by registering its OUI with IEEE and that MAC addresses from the OUI were assigned uniquely. This is for example not necessarily the case for IoT equipment or registrars running in a virtual context in the cloud. IP addresses can be assumed to be unique (enough) when they have global scope or ULA.

When registrar software does not know that no other registrar software or instance of the same software may run on the same host (for example when being packaged as an application), the registrar SHOULD not assume that a host unique name is actually unique, but instead disambiguate it by appending an additional name element to make it unique, such as a process number of the running process.

Picking well-known or unique identifiers for registrar also helps operator to troubleshoot by often eliminating the need to also know the IP addresses associated with the name.

Target host names need to follow the requirements for host names. By those requirements, it is not permitted to use ":" in target host names, for example as part of MAC or IP address based host names. Instance names do not share these syntactical limitation. For operational simplicity, instance names SHOULD be constructed in the same manner as target hostnames in an implementation. For example by replacing ":" with "-".

If the responder needs to indicate different sockets for different (set of) variations, for example, when operating as a Join Proxy, according to Section 3.3.1, then it needs to signal for each socket a separate service instance name with the appropriate port information in its SRV record and the supported variations for that socket in the TXT Record of that service instance name. A responder MAY create the instance and host name for such different variation sockets by appending the variation string to the previously determined instance and host names.

3.5.1.4. Examples

These example use OUI and IPv6 addresses reserved for documentation purposes. Do not re-use these addresses in actual deployments

```
# tBRSKI context
_brski-registrar._tcp.local
                IN PTR  0000-5e00-5314._brski-registrar._tcp.local
0000-5e00-5314._brski-registrar._tcp.local
                IN SRV  1 2 4555 0000-5e00-5314.local
0000-5e00-5314._brski-registrar._tcp.local
                IN TXT  "est-tls" "prm-jose" "cmp"

# cBRSKI
_brski-registrar._udp.local
                 IN PTR  0000-5e00-5314._brski-registrar._udp.local
0000-5e00-5314._brski-registrar._udp.local
                 IN SRV  1 2 5684 0000-5e00-5314.local
0000-5e00-5314._brski-registrar._udp.local
                 IN TXT  "rrm-cose"

# Host name
0000-5e00-5314.local
                IN AAAA  2001:DB8:0815::5e00:5314
```

Figure 2: DNS-SD for single registrar supporting tBRSKI/cBRSKI variations

In example Figure 2, a registrar on a router, that is using mDNS for being discovered supports tBRSKI with "rrm" and "prm" modes across the same TCP socket port 4555, with "est" and "cmp". This leads to the three supported and IANA registry defined variations "est-tls", "prm-jose", and "cmp". For cBRSKI (UDP), it supports the only variation registered through this document, "rrm-cose".

Such a registrar implementation might even support a combination of "prm" with "jose" and "cmp", but at the time of this specification, this exact interoperability aspects of such a combination have at the time of writing of this spec not been investigated and hence it is not listed in the IANA registry. Nevertheless, this may happen later, so it is useful for registrar implementations to allow configuration of variations for its service announcements to allow operational modifications.

This registrar implementation is running on a router that otherwise has no for a host name registered in DNS or DNS-SD, so it is using it’s MAC-address as its target host name, "0000-5e00-5314.local", the same name is used in the registrar service instance names. Running

on a router without modular software, the registrar knows that no other registrar instances can run on the same host and hence the name has no further disambiguating elements.

Note also that there is never a need for two different service instance names between tBRSKI and cBRSKI, because they are distinguished bt the "_tcp" versus "_udp" component of the service instance name.

```
# tBRSKI registrar application
_brski-registrar._tcp.example.org
     IN PTR  noc-registrar-brski-37253._brski-registrar._tcp.example.org
noc-registrar-brski-37253._brski-registrar._tcp.example.org
     IN SRV  1 2 4555 noc-registrar.example.org
noc-registrar-brski-37253._brski-registrar._tcp.example.org
     IN TXT "est-tls" "cmp"

# cBRSKI registrar application
_brski-registrar._udp.example.org
     IN PTR  noc-registrar-cbrski-5376._brski-registrar._udp.example.org
noc-registrar-cbrski-5376._brski-registrar._udp.example.org
     IN SRV  1 2 7533 noc-registrar.example.org
noc-registrar-cbrski-5376._brski-registrar._udp.example.org
     IN TXT "rrm-cose"

# tBRSKI, PRM variation application
_brski-registrar._tcp.example.org
               IN PTR noc-registrar-prm-9735._brski-registrar._tcp.example.org
noc-registrar-prm-9735._brski-registrar._tcp.example.org
               IN SRV 1 2 17355 noc-registrar.example.org
noc-registrar-prm-9735._brski-registrar._tcp.example.org
               IN TXT "prm"

# Host name
noc-registrar.example.org
               IN AAAA  2001:DB8:0815::5e00:5333
```

Figure 3: DNS-SD for a tBRSKI/cBRSKI registrar applications

In the second example Figure 3, a server system in the NOC of customer with domain example.org is set up as the registrar for various BRSKI options. It uses [DNSSD-SRP] to register its DNS-SD names into the example.org domain which it discovers as the default domain. The host name of the server is set to noc-registrar.example.org.

The operator installs three separate registrar applications on this server. One from a vendor whose pledges use tBRSKI, one from an integrator supporting pledges from various "IoT" vendors that use cBRSKI, and one from a manufacturer that has pledges using BRSKI-PRM.

Each of the three applications operates the same way for discovery. It opens a socket for its registrar responder and notes the port number it receives. It determines that SRP is usable, that the default domain is "example.org", and that the host name is noc-registrar. It then forms a unique name from noc-registrar by appending some string abbreviation indicating its mode of operation ("brski", "cbrski", "prm"), and its numeric process identifier - just in case more than one instance of the same application can be started. It then publishes its PTR, SRV and TXT DNS-RR, using these creates unique service instance names, the respective port number in the SRV RR and the variation(s) in the TXT RR.

3.5.2. GRASP

3.5.2.1. Signaling

This document does not specify a mandatory to implement set of signaling options to guarantee interoperability of discovery between initiator and responders when using GRASP. Like for the other discovery mechanisms, these requirements will have to come from other specifications that outline what in [GRASP] is called the "security and transport substrate" to be used for GRASP.

[ACP] specifies one such "security and transport substrate", which is zero-touch deployable. It is mandatory to support for initiators and responders implementing the so-called "Autonomic Network Infrastructure" (ANI). DULL GRASP is used for link-local discovery of proxies, and the ACP is used to automatically and securely build the connectivity for multi-hop discovery of registrars by proxies.

3.5.2.2. Encoding and Examples

To announce protocol variations with [GRASP], the supported Variation is indicated in the objective-value field of the GRASP objective, using the method of forming the Variation String in Section 5.4.4, and listed in the Variation String column of the Table 8 table.

If more than one Variation is supported, then multiple objectives have to be announced, each with a different objective-value, but the same location information if the different Variations can be supported across the same socket because they will all be connected to the same registrar. Different sockets require different objective structures in GRASP anyhow.

Compared to DNS-SD, the choice of encoding for GRASP optimizes for minimum parsing effort, whereas the DNS-SD encoding is optimized for most compact encoding given the limit for DNS-SD TXT records.

```
[M_FLOOD, 12340815, h'fe80000000000000000000000000001', 180000,
    [
     [["AN_Join_Registrar", 4, 255, ""],
      [O_IPv6_LOCATOR,
       h'fe80000000000000000000000000001', IPPROTO_TCP, 4443]],

     [["AN_Join_Registrar", 4, 255, "prm"],
      [O_IPv6_LOCATOR,
       h'fe80000000000000000000000000001', IPPROTO_TCP, 4443]],

     [["AN_Join_Registrar", 4, 255, "rrm"],
      [O_IPv6_LOCATOR,
       h'fe80000000000000000000000000001', IPPROTO_UDP, 4684]]

     [["AN_Join_Registrar_rjp", 4, 255, "rrm"],
      [O_IPv6_LOCATOR,
       h'fe80000000000000000000000000001', IPPROTO_UDP, 4686]]
    ]
]
```

Figure 4: GRASP example for a BRSKI registrar supporting RRM and PRM

Figure 4 is an example for a GRASP service announcement by a registrar in support of BRSKI with both "rrm" and "prm" supported on the same TCP port 4443 and for cBRSKI (COAP over DTLS) on UDP port 4684 in stateful mode and port 4686 for stateless mode . The first variation for "rrm" uses an objective-value of "" for backward compatibility with [BRSKI] where it was introduced. With cBRSKI introducing definitions for the use of GRASP only with this document, this special case is not proliferated, which is why "rrm" is used in the cBRSKI announcements.

Note that one or more complete service instances (in the example 3) can be contained within a single GRASP message without the need for any equivalent to the Service Instance Name of the DNS-SD PTR RR or the Target name of the DNS-SD SRV RR. DNS-SD requires them because its encoding is decomposed into different RR, but it also intentionally introduces the Service Instance Name as an element for human interaction with selection (browsing and/or diagnostics of selection), something that the current GRASP objective-value encoding does not support.

```
[M_FLOOD, 12340815, h'fe80000000000000000000000000001', 180000,
   [
    [["AN_Join_Registrar", 4, 1, ""],
     [O_IPv6_LOCATOR,
      h'fe80000000000000000000000000001', IPPROTO_TCP, 5553]],

    [["AN_Join_Registrar", 4, 1, "prm"],
     [O_IPv6_LOCATOR,
      h'fe80000000000000000000000000001', IPPROTO_TCP, 5555]],

    [["AN_Join_Registrar", 4, 1, "rrm"],
     [O_IPv6_LOCATOR,
      h'fe80000000000000000000000000001', IPPROTO_UDP, 5684]],

    [["AN_Join_Registrar_rjp", 4, 1, "rrm"],
     [O_IPv6_LOCATOR,
      h'fe80000000000000000000000000001', IPPROTO_UDP, 5686]],
   ]
]
```

Figure 5: GRASP example for a BRSKI Join Proxy supporting RRM and PRM

Figure 5 shows a corresponding GRASP service announcement by a Join Proxy that did discover the registrar from Figure 4 and is now announcing the services that it can now proxy. Whereas registrar announcements as in Figure 5 typically use TTL of 255 to be seen across the whole network, Join Proxy announcements are only intended to reach link local neighboring pledges and hence use a TTL of 1.

The use of "" for "rrm" in BRSKI is again for backward compatibility with [BRSKI]. The absence of two announcements for cBRSKI is because there is no stateless mode from Join Proxy to pledge or Registrar-Agent. Instead, the Join Proxy will have have to decide whether to connect to the registrar via stateful or stateless mode, but this decision is invisible on its GRASP announcements.

Noteworthy too is the use of two different ports for "rrm" versus "prm". As the Registrar did announce support for both variations on the same TCP port, the Join Proxy could have done the same, but by using different ports, the Join Proxy can choose independently which Registrar to connect "rrm" versus "prm" sessions to. For example, another Registrar could announce itself for only "prm" and might be preferred by the Join Proxy.

```
[M_FLOOD, 12340815, h'fe800000000000000000000000000001', 180000,
    [["AN_Join_Registrar", 4, 255, "",
     [O_IPv6_LOCATOR,
     h'fe800000000000000000000000000001', IPPROTO_TCP, 4443],
     ["AN_Join_Registrar", 4, 255, "",
     [O_IPv6_LOCATOR,
     h'fe800000000000000000000000000001', IPPROTO_UDP, 4684]]
]

[M_FLOOD, 42310815, h'fe800000000000000000000000000001', 180000,
    [["AN_Join_Registrar", 4, 255, "prm",
     [O_IPv6_LOCATOR,
     h'fe800000000000000000000000000001', IPPROTO_TCP, 44000]]
]
```

Figure 6: GRASP example for a BRSKI Registrar supporting RRM and PRM via separate processes

In Figure 6, a separate application process supports "prm" and hence uses a separate socket with port number 44000 from "rrm", with port 4443. Assuming there is no shared GRASP implementation across the two separate process, such as a separate GRASP process, the announcements from both processes can not be merged into a single GRASP packet. Instead, each one will send its own GRASP announcements separately. This exampe primarily serves as a reminder, that it is necessary for receivers to support receiving multiple announcements from the same sender in GRASP not only within a single packet, but also when they arrive via separate packes. To support implementation cases just as this one.

For a more extensive, DNS-SD compatible encoding of the objective-value that also supports Service Instance Names, see [I-D.eckert-anima-grasp-dnssd].

3.5.3. CORE-LF

3.5.3.1. Overview

"Web Linking", [RFC5988] defines a format, originally for use with HTTP headers, to link an HTTP document against other URIs. Web linking is not a standalone method for discovery of services for use with HTTP.

Based on Web Linking, "Constrained RESTful Environments (CoRE) Link Format", [CORE-LF] introduces a standalone method to discover resources, such as service instances. CORE-LF was introduced primarily for use with [COAP] but it can equally be used for discovery of service instances that use HTTP or any other suitable (web transfer) protocols. This makes CORE-LF an alternative to DNS-SD and GRASP for any of the BRSKI variations.

In CORE-LF, an initiator may use (link-local) IPv6 multicast UDP packet to the COAP port (5683) to discover a possible responder for a requested resource. The responder will reply with unicast UDP. If the IPv6 address of a responder has been configured or is otherwise known to the initiator, it may instead of multicast equally query the parameters of the desired resource via unicast to the default COAP UDP or TCP port (5683).

[RFC9176] defines a "Resource Directory" mechanism for CORE-LF which is abbreviated CORE-RD. Initiators can learn the IPv6 address protocol (TCP or UDP) and port number of a CORE-RD server by some other mechanism (such as DNS-SD) and then use a unicast UDP or TCP COAP connection to the CORE-RD server to discover CORE-LF resources available on other systems. Resource providers can likewise register their resources with the resource directory server using CORE-RD registration procedures.

In summary, CORE-LF including CORE-RD is a mechanism for registration and discovery of resources and hence services which may be preferred in deployments over other options and can equally be applicable to register/discover any variation of BRSKI for any type of BRSKI service.

3.5.3.2. Background

[cBRSKI] specifies the use of CORE-LF as the reference method for pledges to discover registrars - in the absence of any proxies, to allow deployments of scenarios where no proxies are needed - and hence also where [cBRSKI] is not needed. Because BRSKI is designed so that pledges can be agnostic of whether they connect to a registrar directly or via a Join Proxy, the resource/service that the pledge needs to discover is nevertheless called "(BRSKI) Join Proxy (for pleges)", and encoded in CORE-LF as the value "brski.jp" for the resource type attribute ("rt=resource-type") according to [CORE-LF].

The following picture, Figure 7 shows the encoding and an example of this discovery. "ff02::fd" is the link-local scope address for "All Coap Nodes" in IPv6, as introduced in [RFC7390], which also defines IPv6 and site-scoped address options.

Template:

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

RES: 2.05 Content
   <coaps://[Responder_IP_unicast_address]:join-port>; rt="brski.jp"
```

Example:

```
REQ: GET coap://[ff02::fd]/.well-known/core?rt=brski.jp

RES: 2.05 Content
   <coaps://[fe80::c78:e3c4:58a0:a4ad]:8485>;rt=brski.jp
```

Figure 7: CORE-LF discovery of registrar/proxy by pledges

[cPROXY] introduces the operations of a CoAP based Join Proxy both as a connection based Join Proxy as in [BRSKI] (only using UDP connections for COAPs instead of TCP for TLS as in [BRSKI]), but also as a new, stateless Join Proxy - to eliminate the need for potentially highly constrained Join Proxy nodes to keep connection state and avoid the complexity of protecting that state against attacks. The new resource type "brski.rjp" is defined to support stateless Join Proxies to discover registrars and their UDP port number that support the stateless, so-called JPY protocol.

The following picture, Figure 8 shows the encoding and an example of this discovery. [cPROXY] introduces the new scheme "coaps+jpy" for the packet header used by the stateless JPY" protocol. The request in the template is assumed to be based on unicast, relying on another method to discover the IP address of the registrar first. It could equally use COAP site-scoped IP multicast, but in general, the assumption is that registrar will not necessarily be link-local connected to proxies (this may be different in specific deployments). Even though the registrar IP address is hence known, the reply still needs to include this address again because in the [CORE-LF] link format, and [RFC3986], Section 3.2, the authority attribute can not include a port number unless it also includes the IP address.

Template:

```
REQ: GET /.well-known/core?rt=brski.jpy

RES: 2.05 Content
     <coaps+jpy://[Responder_IP_unicast_address]:join-port>;rt=brski.jpy
```

Example:

```
REQ: GET /.well-known/core?rt=brski.jpy

RES: 2.05 Content
     <coaps+jpy://[2001:db8:0:abcd::52]:7633>;rt=brski.jpy
```

Figure 8: CORE-LF discovery of registrars that support stateless JPY protocll by proxies

3.5.3.3. Specification

This section specifies the use of CORE-LF for BRSKI variations. These specifications are backward compatible extensions to what is specified in [cBRSKI] and [cPROXY], except for noted exceptions, where the requirements are narrowed. The following uses terms from the ABNF in section 2 of [CORE-LF] and from [RFC3986] (URI) for explanations and relies on the following template example, Figure 9.

Template:

```
REQ: GET /.well-known/core?rt=brski.*

RES: 2.05 Content
     <scheme://[address]:port path-abempty>;\
       rt=brski-service(;var="brski-variation-string(s)");\
       pw="priority weight"
```

Figure 9: Template for BRSKI discovery with variations

BRSKI responder sockets are indicated in CORE-LF as a URI-Reference. The URI-Reference SHOULD be a URI with a scheme, the IP address of the responder socket and the port used by the responder. It may optionally be followed by a non-empty path-abempty.

URL-references SHOULD not use a domain name instead of an address to allow responders to select a BRSKI responder without requiring DNS support. Likewise, port and scheme MUST be included so that the information can be passed on to consumers without having to modify it. When omitting this information, the full information can only be known in the context of the connections scheme and port through which it was retrieved.

Note that these URL-Reference requirements are stronger than those from [cBRSKI] and [cPROXY] to make extensibility easier.

BRSKI responder sockets MUST include a resource type field indicating a resource type value indicating a BRSKI service, indicated as "brski-service" in Figure 9. This MUST be registered in the IANA "Resource Type Link Target Attribute Values" registry table, and also referenced in the "BRSKI Contexts" registry table Section 5.4.1. A brski-service is a string without "." (single component string).

Discovery of registrar sockets by stateful proxies uses the resource type "brski.rs". This can be used in conjunction with any scheme: https:// for BRSKI and coaps:// for cBRSKI. Stateless registrar sockets use the resource type "brski.rjpy". This currently only support the coaps+jpy:// scheme. By its nature, it can only be used with schemes that rely on UDP. These resource type uses are no change over [cBRSKI] and [cPROXY]. This document does not specifiy how to discover BRSKI-PLEDGE via CORE-LF.

The variations supported by a BRSKI responder socket are indicated via the optional "var=" link-extension. The value is a quoted-string of one or more space contatenated BRSKI variation strings. The absence of a "var=" link-extension indicates support for only the default variation for the BRSKI context to which the BRSKI service belongs. This can also be indicated as "var=".

The optional "pw" target attribute indicates priority and weight for the selection of the resource target with the semantic and format defined in [RFC2782] for priority and weight in DNS SRV resource records. If the attribute pw is absent, then it is assumed to mean pw="65535 0".

A non-empty path-abempty indicates a path prefix for the endpoints supporting the BRSKI service and variation that is shorter than the default endpoint paths specified for the service.

3.5.3.4. Examples

```
REQ: GET /.well-known/core?rt=brski.*

RES: 2.05 Content
Content-Format: 40
Payload:
 <https://[2001:DB8:0815::5e00:5314]:4555>;        # [1]
        rt=brski.rs;var="est-tls prm-jose cmp";
        pw="1 2",
 <https://[2001:DB8:0815::5e00:5314]:4555>;        # [2]
        rt=brski.jp;var="est-tls prm-jose cmp";
        pw="1 2",
 <coaps://[2001:DB8:0815::5e00:5314]:5684/b>;      # [3]
        rt=brski.rs;var=,
        pw="1 2",
 <coaps://[2001:DB8:0815::5e00:5314]:5684/b>;      # [4]
        rt=brski.jp;var=,
        pw="1 2",
 <coaps+jpy://[2001:DB8:0815::5e00:5314]:6534/b>;  # [5]
        rt=brski.rjpy;var=,
        pw="1 2"
```

Figure 10: CORE-LF examples for BRSKI variations

Figure 10 shows example BRSKI variations in CORE-LF format. Note that the example is pretty-printed through indentation and breaking long lines. This additional white space is not compatible with actual CORE-LF output. Likewise, the text following "#" are editorial comments.

Example [1] is the equivalent announcement for a BRSKI registrar service as shown for DNS-SD in Figure 2 except for the absence of any service instance. Note the use of "var=" to indicate the list of variation strings supported and "pw=" to indicate priority and weight as in DNS-SD.

[3] is likewise the comparable example for the cBRSKI registar example with DNS-SD. Note that here, a non-empty path-abempty "/b" is used to indicate a shortened endpoint prefix path for the service. There is no equivalent in DNS-SD defined. When discovering a service via DNS-SD, the service will need to use the (longer) pre-defined endpoint prefixes, such as "/brski" and "/est" instead of "/b".

Example [2] is the same socket as [1], but announced as a Join Proxy socket for pledges. Likewise, [4] is the same socket as [2] announced as a Join Proxy socket for pledges. Finally, [5] announces the registrars socket in support of stateless Join Proxies using the JPY header encapsulation.

3.5.3.5. Resource Type Considerations

CORE-LF expresses information about resources of a target identified by a resource type. This specification encodes BRSKI services in CORE-LF also as a resource types, as specified in Section 3.5.3.3. For the purpose of CORE-LF, a BRSKI service is just another resource, except that it characterizes the overall functionality available across a connection to the target, composed of a sequence of endpoint instantiations. In addition, this behavior is further refined by the list of supported variations indicated.

Often, resources in CORE-LF do - instead of a service - describe details of as little as a single endpoint, such as its URL prefix and format encoding. The reason why this fine-grained specification is not a good replacement for the concept of service and variation is that the avilability of a set of endpoints with specific encodings does not imply whether the target does support the desired specific sequencing of instantiating those endpoints, including the use of any endpoint encoding option in any combination.

Making such arbitrary combinations a requirement can easily lead to more generic, but also more costly implementations and testing requirements without necessarily gaining deployment benefit.

BRSKI resource types which are not treated as services according to this specification can still be used if so desired to amend the discovery of shortened endpoints, as shown in Figure 11.

```
RES: 2.05 Content
Content-Format: 40
Payload:
 <https://[2001:DB8:0815::5e00:5314]:4555/b>;      # [1]
        rt=brski.rs;var="est-tls prm-jose cmp";
        pw="1 2",
 <https://[2001:DB8:0815::5e00:5314]:4555/b/rv>;   # [2]
        rt=brski.rs.requestvoucher,
 <https://[2001:DB8:0815::5e00:5314]:4555/b/vs>;   # [3]
        rt=brski.rs.voucher_status,
 </b/rv>;rt=brski.rs.rv,                           # [4]
 </b/vs>;rt=brski.rs.vs,                           # [5]
```

Figure 11: CORE-LF resource examples

[1] shows how the prefix for all BRSKI endpoints over "https://" can be shortened from "/.well-known/brski" to "/b". Nevertheless, this would still make it necessary to use "/b/requestvoucher" and "/b/voucher_status" as endpoints.

[2] and [3] show how to shorten those two endpoints to "/b/rv" and "/b/rs" by creating resource types "brski.rs.rv" and "brski.rs.vs". By using resource type prefix "brski.rs." for both of them as well as path prefix "/b", it can be implied that these endpoints are part of the service specified in [1],

These discovery options can be further compacted such as shown in example [4] and [5] when assuming that the abbreviations "rv" and "vs" are also known even by BRSKI implementations from [cBRSKI]. Likewise, the full socket details can be avoided when one can infer it from context.

While these shortenings can be highly useful in often called resources, each endpoint in BRSKI is typically only instantiated once by a pledge, so the overall savings in communication data becauseof these shortenings is likely negligible, and it is better to define short endpoint paths into the variation specification if they are likely needed, such as done in [cBRSKI], such that it is not necessary in cBRSKI to add such shortenings in discovery. For these reasons, this document does not specify if or how to use such resource targets in cunjunction with BRSKI discovery but only discusses possibilities and limitations here.

Considerations for such non-service resource type use in BRSKI nevertheless introduces one requirement to avoid conflicts: The names of BRSKI services MUST not duplicate the endpoint names of any resources specified for BRSKI protocols. This means that "rv" or "vs" can not be used to create BRSKI service name resource types "brski.rv" or "brski.rs", and likewise, additional BRSKI endpoints can not be called "rs", "jp", "jpy" or any other string registered in the BRSKI discover registry tables.

4. Updates

4.1. Updates to RFC9733

This document updates [BRSKI-AE], section 5.1 with the following text which is to be read as if appended to the end of that section.

Instead of using the minimalist "brski-reg-cmp" specified in [BRSKI-AE], this document RECOMMENDS for new implementations of BRSKI with CMP and DNS-SD discovery to use the signaling elements specified in this document with the following benefits.

For DNS-SD, the equivalent for "brski-reg-cmp" is "brski-registrar" (see {#subreg-service-names}) with the TXT string "cmp" (see Section 5.4.4). This automatically allows to use Join Proxies supporting this specification. They will use "brski-proxy" with TXT

string "cmp" to indicate their support to transparently proxy also for a CMP supporting registrar. Note that this will allow to use CMP in cBRSKI as both "brski-registar" and "brski-proxy" are also registered for use with UDP.

Likewise, "brski-registrar-rjp" with TXT string "cmp" and UDP can be used to support CMP with stateless Join Proxies supporting this specification and the registrations in {#subreg-service-names} allow to discover Registrars / Proxies for BRSKI and cBRSKI with CMP also with GRASP and CORE-LD Discovery Mechanisms.

If backward compatibility is required, "brski-reg-cmp" can continue to be announced by registrars unchanged.

4.2. Updates to BRSKI-PRM

This documents updates [BRSKI-PRM] by adding the following text to the end of section 6.1.1.

With the procedures specified in this document, Registrar Agents can discover Registrars supporting PRM by discovering Registrars that announce a variation which includes "prm". Because [BRSKI-PRM] specifies the use of JOSE for voucher encoding, the correct Variation String to discover is "prm-jose", as also registered in Section 5.4.4. Discovery can use DNS-SD, CORE-LF or GRASP using the encodings specified in this document and registered in Section 5.4.4 for the Variation String and Section 5.4.2 for the Service Name.

Registrar Agements MAY use Join Proxies supporting the procedures of this document to reach Registrars supporting PRM andusing the procedures of this document. This may be of interest if the network available in the location where the Registrar Agent is operating to access the Pledge is not fully trusted but Join Proxies are used to only allow connections to Registrars.

This documents updates [BRSKI-PRM] section 6.1.2 with the procedures specified in Section 3.4. Those procedures are meant to be fully backward compatible with those specified in [BRSKI-PRM], but adds more details and suggestions for encoding of signaling elements.

4.3. Updates to RFC6690

This document adds the text of Section 4.3.1 to [CORE-LF]. It recommenda to IANA to document that addition as indicated in Section 5.1.1.

4.3.1. Additional Resource Type requirements for "brski"

The following text is to be read as if inserted into [CORE-LF] section 7.4 after the second paragraph.

For the Resource Type (rt=) Link Target Attribute table, values starting with the characters "brski" are also subject to the following paragraph requirements.

Resource Type values with the "brski" prefix SHOULD support BRSKI discovery as specified in ([ThisRFC]). The specification of such a Resource Type SHOULD NOT prohibit or conflict with the ability to combine variation type values as established by [ThisRFC]. The specification SHOULD include or point to a definition how the Resource Type is to be included into the "BRSKI Context Registry Table" Section 3.1.7. Review of these requirements is to be coordinated between Designated Experts for the Core parameters (core-parameters@ietf.org) and those for the BRSKI Discovery Parameters Registry (Section 5.4).

5. IANA considerations

5.1. Core Parameters

5.1.1. Resource Type Link Target Attribute Values

This document introduces additional requirements for the registration of Core Resource Type values with prefix "brski" into the "Resource Type (rt=) Link Target Attribute Values" sub-registry of the "Constrained RESTful Environments (CoRE) Parameters" registry, as specified in Section 4.3.1.

IANA is suggested to document these additions by adjusting the registration procedure table for the "Resource Type (rt=) Link Target Attribute Values" sub-registry as shown in Table 1.

Range	Registration Procedures	Note
value starts with "core"	IETF Review	
value starts with "brski"	Specification Required	Additional requirements in ThisRFC Section 4.3.1
all other values	Specification Required	

Table 1: Suggested updated registration procedure table for Core (rt=) Rink Target Ranges

5.1.2. Target Attributes

Attribute Name	Brief Description	Change Controller	Reference
var	List of supported variations of target	IETF	[ThisRFC]
pw	DNS SRV compatible priority and weight of resource target	IETF	[ThisRFC]

Table 2: Target Variation and Priority/Weight Attributes

IANA is asked to add entries for "var" and "pw" according to above Table 2 to the "Target Attributes" table.

The "var" target attribute is meant to be used for BRSKI targets as specified in this document. It is also meant to be usable for other targets if so desired - to indicate variations of the resource type of the target. For targets with a non-BRSKI resource target (not using "rt=brski.*"), the format of the value may be different than specified for BRSKI.

The "pw" target attribute indicates priority and weight for the selection of the resource target with the semantic and format defined in [RFC2782] for priority and weight in DNS SRV resource records. If the attribute pw is absent, then it is assumed to mean pw="65535 0".

5.2. Service Names Registry

IANA is asked to modify and amend the "Service Name and Transport Protocol Port Number Registry" registry (https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.txt) as follows.

* brski-proxy, brski-registrar and brski-registrar-rjp are to be added as Service Names for the "udp" protocol using ThisRFC, Section 5.4.2 as the reference.

* The registrations for brski-proxy and brski-registrar for the "tcp" protocol are to be updated to also include ThisRFC, Section 5.4.2 as their reference.

* The Defined TXT keys column for brski-proxy, brski-registrar and brski-registrar-rjp for "tcp" and "udp" protocols are to state the following text:

  Defined TXT keys: <variation-string(s)>, See ThisRFC Section 3.5.1.2, Section 5.4.4

5.3. GRASP Objective Names Registry

IANA is asked to add the following entry to the "GeneRic Autonomic Signaling Protocol (GRASP) Parameters Registry" page, "GRASP Objective Names Registry".

Objective Name	Reference
AN_join_registrar_rjp	ThisRFC, Section 5.4.2

Table 3: GRASP Objective Names Registry addition

5.4. BRSKI Discovery Parameters

IANA is asked to add a registry called "BRSKI Discovery Parameters" to the "Bootstrapping Remote Secure Key Infrastructures (BRSKI) Parameters" registry webpage (https://www.iana.org/assignments/brski-parameters/brski-parameters.xhtml).

The BRSKI Discovery Parameters registry includes several sub-registries that depend on each other. Due to the requirement of an IANA registry with sub-registries, one table has to be choosen to represent the overall registry. The table used to represent the BRSKI Discover Parameters is the list of discovery mechanisms supported.

The Registration Procedure for this registry is Specification Required, where Expert Review has to ensure compliance of entries with the following rules.

Additions to this registry require a specification of the use of the newly registered discovery mechanism with BRSKI discovery procedures in the way they are specified in this document for DNS-SD, GRASP and CORE-LF.

Changes/extensions to the procedures for any existing registered discovery mechanism, including DNS-SD, GRASP or CORE-LF can be done by adding such specification to the Reference(s) colum.

Any incremental changes to a discovery mechanism procedures require the same or higher level of specification as the first one introducing the discovery mechanism. Changes to the procedures for DNS-SD, GRASP, CORE-LF do hence require an IETF standards track specification for updates.

The initial content is as follows.

Designated Experts: TBD.

Registration Procedure: Specification Required.

Notes: See [ThisRFC], Section 5.4

Discovery Mechanism	Reference(s)	Notes
DNS-SD	ThisRFC	
GRASP	ThisRFC	
CORE-LF	ThisRFC	

Table 4: Discovery Mechanisms initial registry table

5.4.1. Contexts

IANA is asked to create a sub-registry called "Contexts" in the "BRSI Discovery Parameters" registry.

The Registration Procedure for this registry is Specification Required, where Expert Review has to ensure feasibility of entries with the following behavior/rules.

A Context is a name for a group of one or more BRSKI services. This sub-registry register a Contexts name and the list of Variation Types defined for the Context.

Variation Types are case independent consisting only of the letters a-z and the digits 0-9, must start with a letter and be at most 12 characters long.

Registration of new Contexts MUST include a specification of how to use discovery with the new Context using the specifications for BRSKI, cBRSKI and BRSKI-PLEDGE in this document as examples.

Technically, Contexts with the same set of Variation Types exist for example to allow registering and hence discovering different protocol stacks for otherwise logically the same type of services, or else a BRSKI pledge might be discovering a cBRSKI registrar. See Section 5.4.2 for more details.

The order of Variation Types defines the order in which Variation Type Values are concatenated to generate Variation Strings as described in Section 5.4.4. Therefore Variation Types ones registered SHOULD NOT be changed or deleted so that new Variation Strings will be constructed consistently with old ones. Only new Variation Types may be appended through regisrtration updates.

Registration of additional Variation Types MUST include a specification of how they map to specific function of the services that use them as this specification does for BRSKI, cBRSKI and BRSKI-PLEDGE.

Registrations of a new Context MAY initially not define any Variation Types if Variations are not yet considered for a Context.

The initial content of this sub-registry is as follows.

Designated Experts: TBD.

Registration Procedure: Specification Required.

Notes: See [ThisRFC], Section 5.4.1

Context	Variation Types	Reference(s)	Notes
BRSKI	mode, vformat, enroll	ThisRFC	
cBRSKI	mode, vformat, enroll	ThisRFC	
BRSKI-PLEDGE	mode, vformat, enroll	ThisRFC	

Table 5: Contexts initial sub-registry table

5.4.2. Service Names

IANA is asked to create a sub-registry called "Service Names" in the "BRSI Discovery Parameters" registry.

The Registration Procedure for this registry is Specification Required, where Expert Review has to ensure feasibility of entries with the following behavior/rules.

Each entry registers a Service Name intended for discovery of a BRSKI related service using a specific Discovery Mechanism. The service is to be accessed via specific service (transport) stack Parameter(s) that are required to be known by the discovery mechanism.

Because they may be used in the future in protocol encodings, Variation Type Choices SHOULD be as simple as possible. They SHOULD consist only of the letters a-z and the digits 0-9, SHOULD start with a letter and be at most 12 characters long.

Depending on the discovery mechanism, the Service Name represents a different signaling element of the discovery mechanism as follows.

o For DNS-SD, the tables Service Name is the DNS-SD Service Name which MUST also be registered in the IANA "Service Name and Transport Protocol Port Number Registry" for use with the DNS-SD protocol indicated in the Parameter(s) column.

o For GRASP, the Service Name is the GRASP Objective Name which MUST also be registered in the IANA "GeneRic Autonomic Signaling Protocol (GRASP) Parameters" / "GRASP Objective Names" registry.

o For CORE-LF, the Service Name is the CoRE Resource Type (rt=) value, which MUST be registered in the IANA "Constrained RESTful Environments (CoRE) Parameters" / "Resource Type (rt=) Link Target Attribute Values" registry.

Context is a grouping of one or more services with the same set of Variation Types and Values as defined below (Section 5.4.1). Re-use of the same Service Name across multiple Contexts requires distinction of the services through Parameter(s) that are used as additional distinguishers (beside the Service Name) in the discovery mechanism.

For example, AN_Proxy is used as the GRASP Objective (aka: "Service Name") to discover [BRSKI] BRSKI Join Proxies and connect to them via TCP. This is indicated through the IPPROTO_TCP parameter in GRASP. Likewise, when a constrained BRSKI proxy as described in [cPROXY] is to be discovered via GRASP, IPPROTO_UDP is to be used in GRASP to distinguish such a cBRSKI Proxy from a BRSKI Proxy.

Informal names for services are best documented in the Notes column.

The initial registrary table adds to the registrations from other BRSKI RFCs new registrations to discover BRSKI Registrar and Join Proxy via CORE-LF. These do not require new registrations in the CORE registry because those registrations do not distinguish on the transport stack, and hence the prior CORE registrations for cBRSKI are equally applicable to BRSKI. In addition, the registrations to discover cBRSKI stateful and stateless Registrars and Join Proxies via GRASP and DNS-SD are included. The necessary additional registrations for these are included in Section 5.2 for DNS-SD and Section 5.3 for GRASP. With these registrations, both BRSKI and cBRSKI services can use DNS-SD, GRASP or CORE-LF for discovery.

The initial content of this sub-registry is as follows.

Designated Experts: TBD.

Registration Procedure: Specification Required.

Notes: See [ThisRFC], Section 5.4.2

Service Name(s)	Context	Discovery Mechanism	Parameter(s)	Reference(s)	Notes
brski.jp	BRSKI	CORE-LF	https	THIS-RFC	Proxy
brski.rs	BRSKI	CORE-LF	https	THIS-RFC	Registrar

					(stateful)
brski-proxy	BRSKI	DNS-SD	tcp	RFC8995	Proxy
brski-registrar	BRSKI	DNS-SD	tcp	RFC8995	Registrar (stateful)
AN_Proxy	BRSKI	GRASP	IPPROTO_TCP	RFC8995	Proxy
AN_join_registrar	BRSKI	GRASP	IPPROTO_TCP	RFC8995	Registrar (stateful)
brski.jp	cBRSKI	CORE-LF	coaps	I-D.ietf-anima-constrained-voucher	Proxy
brski.rs	cBRSKI	CORE-LF	coaps	I-D.ietf-anima-constrained-join-proxy	Registrar (stateful)
brski.rjp	cBRSKI	CORE-LF	coaps	I-D.ietf-anima-constrained-join-proxy	Registrar (stateless)
AN_Proxy	cBRSKI	GRASP	IPPROTO_UDP	THIS-RFC	Proxy
AN_join_registrar	cBRSKI	GRASP	IPPROTO_UDP	THIS-RFC	Registrar (stateful)
AN_join_registrar_rjp	cBRSKI	GRASP	IPPROTO_UDP	THIS-RFC	Registrar (stateless)
brski-proxy	cBRSKI	DNS-SD	udp	THIS-RFC	Proxy
brski-registrar	cBRSKI	DNS-SD	udp	THIS-RFC	Registrar (stateful)
brski-registrar-rjp	cBRSKI	DNS-SD	udp	THIS-RFC	Proxy (stateless)
brski-pledge	BRSKI-PLEDGE	DNS-SD	tcp	I-D.anima-brski-prm, THIS-RFC	

Table 6: Service Name(s) initial sub-registry table

5.4.3. Variation Type Choices

IANA is asked to create a sub-registry called "Variation Type Choices" in the "BRSI Discovery Parameters" registry.

The Registration Procedure for this registry is Specification Required, where Expert Review has to ensure feasibility of entries with the following expectations/rules.

Each row registers one Variation Type Choice for one Context (as registered in Section 5.4.1) and specifies the Variation Type (also as registered in Section 5.4.1) for which it is a choice.

All Variation Type Choices MUST be unique across all Variation Types so that the Variation Type can always be deduced from the Variation Type Choice, permitting future encodings that do not necessarily have to use the Variation String representation (as registered in Section 5.4.4), and also allowing to recognize erroneous variation repesentations easier.

Because they are used in protocol encodings, Variation Type Choices MUST be as simple as possible. They MUST consisting only of the characters a-z and 0-9.

The "Dflt" flag indicates that the Variation Type Choice is the default choice for its Variation Type. Default choices are not included in the Variation String encoding of Variations. When new Variation Types are added to a Context to introduce variations of a new aspect of BRSKI, then the choice that is backward compatible has to become the Default choice for that new Variation Type.

The "Rsvd" flag indicates a choice that has not sufficiently been vetted/specified to allow its use in Variations (Section 5.4.4, but that is known to be a possible candidate and is reserved to track that possible extension through future specification work.

References in parenthesis specify the necessary functionality for a Variation Type Choice but do not specify the actual registration of the Veriation Type Choice.

The initial content is as follows.

Designated Experts: TBD.

Registration Procedure: Specification required.

Flags	Variation Type Choice	Variation Type	Context	Reference(s)	Note(s)
Dflt	rrm	mode	BRSKI	(RFC8995),ThisRFC	Registrar Responder Mode
	prm	mode	BRSKI	(I-D.ietf-anima-brski-prm),ThisRFC	Pledge Responder Mode
Dflt	cmsj	vformat	BRSKI	(RFC8368,RFC8995),ThisRFC	CMS-signed JSON Voucher
	jose	vformat	BRSKI	(I-D.ietf-anima-jws-voucher),ThisRFC	JOSE-signed JSON
	cose	vformat	BRSKI	ThisRFC	CBOR with COSE signature
Dflt	est	enroll	BRSKI	(RFC7030,RFC8995),ThisRFC	Enrollment over Secure Transport
	cmp	enroll	BRSKI	(RFC9733,RFC9483),ThisRFC	Lightweight CMP Profile / BRSKI-AE
Rsvd	scep	enroll	BRSKI	ThisRFC	common legacy option
Dflt	rrm	mode	cBRSKI	(I-D.anima-constrained-voucher),ThisRFC	Registrar Responder Mode
Dflt	cose	vformat	cBRSKI	(I-D.ietf-anima-constrained-voucher),ThisRFC	CBOR with COSE signature
	cmsj	vformat	cBRSKI	ThisRFC	CMS-signed JSON Voucher

	jose	vformat	cBRSKI	ThisRFC	JOSE-signed JSON
Dflt	est	enroll	cBRSKI	(RFC9148), ThisRFC	Enroll via EST over COAP
	cmp	enroll	cBRSKI	(RFC9733,RFC9483),ThisRFC	Lightweight CMP Profile / BRSKI-AE
Dflt	prm	mode	BRSKI-PLEDGE	(I-D.ietf-anima-brski-prm),ThisRFC	Pledge responder Mode
Dflt	jose	vformat	BRSKI-PLEDGE	(I-D.ietf-anima-brski-prm),ThisRFC	JOSE-signed JSON
Rsvd	cmsj	vformat	BRSKI-PLEDGE	(I-D.ietf-anima-brski-prm),ThisRFC	CMS-signed JSON Voucher
Rsvd	cose	vformat	BRSKI-PLEDGE	(I-D.ietf-anima-brski-prm),ThisRFC	CBOR with COSE signature
Dflt	est	enroll	BRSKI-PLEDGE	(I-D.ietf-anima-brski-prm),ThisRFC	Enroll via EST modified for PRM
	cmp	enroll	BRSKI-PLEDGE	(RFC9733,RFC9483,I-D.ietf-anima-brski-prm),ThisRFC	Lightweight CMP Profile with PRM

Table 7: Variation Type Choices initial sub-registry table

5.4.4. Variations

IANA is asked to create a sub-registry called "Variations" in the "BRSI Discovery Parameters" registry.

The Registration Procedure for this registry is Specification Required, where Expert Review has to ensure feasibility of entries with the following behavior/rules.

Each row registers one or more Variation String(s) that are representing one Variation in one Context for and specifies the combination of Variation Type Values indicated by that Variation.

Variation Type Values MUST be listed in the order in which they appear in Section 5.4.1, including the default Variation Type Values for the Variation Types known at the time of registration. Once a Variation String for a Context is registered, its listed Variation Type Values SHOULD NOT be modified (except for erroneous registration).

The primary Variation String SHOULD be derived by concatening the Variation Type Values listed, concatenated by "-", except for default values. This is called the standard Variation String construction rule. The empty string is represented as "".

Any Variation String not meeting that construction scheme SHOULD receive a note explaining why.

When later potentially new Variation Types are introduced into the XXX table, then they will not lead to a change in the Variation String(s), instead it is understood that the Variation will represent the default Variation Type Values for any such additional Variation Types.

This registry table exists to document which specification(s) utilize specific Variations so that implementations do not have to consider supporting arbitrary possible (but not validated by specification) Variations.

The subregistries specified below register the permissible Context and Variation Type Values.

The initial content is as follows.

Designated Experts: TBD.

Registration Procedure: Specification required.

Notes: (1) The Variation String "EST-TLS" is equivalent to the Variation String "" and is required and only permitted for the AN_join_registrar objective value in GRASP for backward compatibility with [BRSKI], where it is used for this variation. Note that AN_proxy uses "".

Variation String	Context	Variation	Specification(s)	Notes
"" / "EST-TLS"	BRSKI	rrm cmsj est	RFC8995	See Note (1)
cmp	BRSKI	rrm cmsj cmp	RFC9733	
prm-jose	BRSKI	prm jose est	(I-D.ietf-anima-jws-voucher,I-D.ietf-anima-brski-prm),ThisRFC	I-D.ietf-anima-brski-prm also includes required extensions to EST
""	cBRSKI	rrm cose est	I-D.ietf-anima-constrained-voucher	
prm-jose	BRSKI-PLEDGE	prm jose est	I-D.ietf-anima-brski-prm	

Table 8: Variations initial sub-registry table

5.5. BRSKI Well-Known URIs fixes (opportunistic)

The following change requests to "https://www.iana.org/assignments/brski-parameters/brski-parameters.xhtml#brski-well-known-uris" are cosmetic in nature and are included in this document solely because support for Endpoint URIs is implied by the mechanisms specified in this document and the existing registry has these cosmetic issues.

1. IANA is asked to change the name of the first column of the table from "URI" to "URI Suffix". This is in alignment with other table columns with the same syntax/semantic, such as "https://www.iana.org/assignments/well-known-uris/well-known-uris.xhtml".

2. IANA is asked to change the Reference from "RFC8995" to "RFC8995, Section 8.3.1".

3. IANA is asked to include the following "Note" text: The following table contains the assigned BRSKI protocol Endpoint URI suffixes under "/.well-known/brski"." - This note is added to introduce the term "Endpoint" into the registry table as that is the term commonly used (instead of URI) in several of the memos for which this discovery document was written. It is meant to help readers map the registry to the terminology used in those documents.

6. Security Considerations

In [BRSKI-PRM], pledges are easier subject to DoS attacks than in [BRSKI], because attackers can be initiators and delay or prohibit enrollment of a pledge by opening so many connections to the pledge that a valid Registrar-Agents connection to the pledge may not be possible. Discovery of the pledge via DNS-SD increases the ability of attackers to discover pledges against which such DoS attacks can be attempted.

Especially when supporting DNS-SD browsing across unicast DNS, pledges MUST implement DoS prevention measures, such as limiting the number and rate of accepted TCP connections on a per-initiator basis. If feasible for the implementation, simultaneous connections SHOULD be possible, so that an ongoing attacker connection will not delay a valid Registrar-Agent connection. When accepting connections, a strategy such as LRU MAY be used to ensure that an attacker will not be able to monopolize connections.

Browsing via DNS-SD, especially via unicast DNS which makes information available network-wide does also introduce a perpass attack, gathering intelligence against what type and serial number of devices are installed in the network. Whether or not this is seen as a relevant risk is highly installation dependent. Networks SHOULD implement filtering measures at mDNS and/or DNS RR/services level to prohibit such data collection if there is a risk, and this is seen as an undesirable attack vector.

Service instance names as defined in Section 3.4 are used to discover pledges by their manufacturer assiged serial numbers. Today, DNS-SD does not provide security against impersonation of such service instance names. Instead, impersonation can and will only be discovered after performing BRSKI connections to the pledge. It should be noted, that the scheme used by Section 3.4 could actually be used to protect against impersonation when [DNSSD-SRP] with some security extension is used: Pledges need to signal their IDevID for their SRP TLS connection, and the SRV server needs to have the same manufacturer Service Instance Name schema and manufacturer trust anchor information as BRSKI registrars and can then allow only the permissible service instance name DNS-SD RRs for this pledge. In

fact, the SRP server could create the all necessary Section 3.4 required DNS-SD RRs from the IDevID information even if the pledge itself is not requesting them or is requesting other DNS-SD RRs. Definition of these procedures is outside the scope of this specification though.

None of the discovery mechanisms (DNS-SD, GRASP or CORE-LF) are necessarily secure. Instead, it is a matter of their deployment, how trustworthy service announcements are. In an unprotected deployment with DNS-SD via mDNS for example, an attacker could attract connections from responders by announcing itself with the best priority value. When a deployment instead uses a secure domain deployment model, such as an [ACP], a secured wireless mesh technology, or discovery via a secured DNS server, then service announcements are typically assumed to be trustworthy and with them their service parameters. In those deployments, the security question is then primarily the attack vectors to impair such a responder to make it behave in undesirable means.

7. Acknowledgments

Many thanks for reviews by Arthur Hecker, Steffen Fries and discussions/feedbacks by Brian Carpenter, Michael Richardson, Michael Kovatsch. Special thanks to Amanda Barber for her IANA section review.

7.1. Change log

[RFC Editor: please remove this section.]

WG draft 09:

Small editorial fix. Table incorrectly split by empty lines.

WG draft 08:

Overhauled IANA section after IANA early review. Changed tables so every table has only one key registered item which is usually in the first colum. Eliminated formatting based registration, eg.: no more "multi-line" registration - does notwork because the registries are databased. Removed optical beautification of keeping cells empty to indicate repetition of previous row content. This too does not work for IANA registry because they allow re-ordering of rows by column.

Also rewrote/shortened the Registry Data Model section to fit the new IANA section. Put Discussios section before IANA discussion so it is clear it is part of the overall data model, not the IANA representation.

Added updates to BRSKI-AE and BRSKI-PRM text that details how BRSKI Discovery is to be used for both RFCs mechanisms. Also made this doc an update to BRSKI-AE to ensure it is tracked that way.

Moved all other IANA consideration before the ask for the new BRSKi Discover Parameters, as this looks more logical - except for the opportunistic ask.

In result of creating more tables and hence removing colums, the tables "should" actually fit into the 72 character TXT rendering as soon as the I-D... draft references in them are replaced by RFC references, which should happen in RFC editor queue before this document gets published (hope, hope...).

Removed appendix "possible future variations"

Made document update to rfc6690 to allow formalizing request to update CORE RT= parameter range table with BRSKI entries, see section 4.1.1.

WG draft 07:

Defined document to be update to draft-ietf-anima-brski-prm (for the specifiation of IDevID derived DNS-SD discovery of pledges)

Resolved open Q text wether SRP allows discovery of SRP server. According to Stuart Cheshire this is supported.

Incorporated initial Feedback from Michael Kovatsch

Added section explaining that there is no spec for how to do pledge discovery via CORE-LF or GRASP.

* Rewrote 2.1 Challenges to now be a more comprehensive set of 3 example deployment issues without this work.

Incorporated review by Artur Hecker

* Rewrote abstract to more comprehensively (and easier understandable) describe the scope of this document

* Simplified terminology: removed "variation context", not only calling it "context".

* changed name of BRSKI context in IANA registry to ’tBRSKI’ (TCP BRSKI) - to make it easier to know when text refers to BRSKI as an overall cncept or specifically to the TCP based set of variations of BRSKI.

* Removed duplicate text paragraphs in proxy sections.

* Added note about (in)security of discovery mechanisms in general and how deployments typically defer to the "secure domain" context to overcome this issue.

* Large number of textual fixes (thanks a lot for the thorough read!).

WG draft 06:

Initial overview review feedback from Michael Kovatch and Artur Hecker

Made abstract and Challenges introduction hopefully better explaining the scope of the document and motivate it’s need.

Review Steffen Fries:

Cleaned up terminology IP/IPv6 -> IP/IPv4/IPv6.

CA -> trust anchor

BRSKI proxy -> Join Proxy for consistency with RFC8995.

Added mentioning of Registrar-Agent from BRSKI-PRM where appropriate

Rewrote 2.1.3 to make the functionalty of variation agnostistic proxying clearer (hopefully)

Rewrite 3.1.1 to clearer define role and services and distinguish them.

Changed "cms" to "cmsj"(as well as derived variation strings) so that it is clear that this variation type does not mean all possible encoding options for CMS but only JSON.

Added explanation about the fact that a variation may introduce changes to a variation type component that shares the same name (3.1.6)

Noted that discovery of pledges does not apply in 3.2 which talks about redundant service discovery/selection.

removed last paragraph from 3.3.2.2 - duplicate from earlier section.

Improved 3.4.3 by better structuring the example figure and rewriting the explanation text as a step-by-step explanation how a Registrar-Agent would perform the steps.

Fixed small bugs in GRASp example section 3.5.2.2 but ended up improving examples a lot and make them more useful (registrat AND proxy)

Still can not figure out how to nicely get hotlinks to the terminology section definitions. They just show up in text format as "Section 1" or the like. Giving up on the idea. TBD: Maybe ask RFC editor/RSWG. Likewise, i can not have references to BRSKI RFCs both with a logical name like BRSKI-PRM and in other places references with the RFC number. I need to define both as references and then the same RFC will have two entries. Stupid. Now i only have logical references, but in all places where i need to actually reference the RFC by number, such as IANA registries or pictures, i do not use references anymore.

WG draft 05:

Mayor update to specifiy resilience aspects in selection of responders.

Mayor update/simpliciation of CORE-LF section.

WG draft 02/03:

Fix up tables to be correctly rendered by html output.

WG draft 01:

Core-LF improvements / interim work.

WG draft 00:

Added section for CORE-LF. Still missing to update existing text with the CORE-LF definitions.

Individual version 01:

Various enhancements

Individual version 00:

Initial version.

8. References

8.1. Normative References

[ACP] Eckert, T., Ed., Behringer, M., Ed., and S. Bjarnason, "An Autonomic Control Plane (ACP)", RFC 8994, DOI 10.17487/RFC8994, May 2021, <https://www.rfc-editor.org/rfc/rfc8994>.

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

[BRSKI-AE] von Oheimb, D., Ed., Fries, S., and H. Brockhaus, "BRSKI with Alternative Enrollment (BRSKI-AE)", RFC 9733, DOI 10.17487/RFC9733, March 2025, <https://www.rfc-editor.org/rfc/rfc9733>.

[BRSKI-PRM] Fries, S., Werner, T., Lear, E., and M. Richardson, "BRSKI with Pledge in Responder Mode (BRSKI-PRM)", Work in Progress, Internet-Draft, draft-ietf-anima-brski-prm-23, 3 June 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-anima-brski-prm-23>.

[cBRSKI] Richardson, M., Van der Stok, P., Kampanakis, P., and E. Dijk, "Constrained Bootstrapping Remote Secure Key Infrastructure (cBRSKI)", Work in Progress, Internet-Draft, draft-ietf-anima-constrained-voucher-29, 18 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-anima-constrained-voucher-29>.

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

[cPROXY] Dijk, E., Richardson, M., Van der Stok, P., and P. Kampanakis, "Join Proxy for Bootstrapping of Constrained Network Elements", Work in Progress, Internet-Draft, draft-ietf-anima-constrained-join-proxy-18, 19 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-anima-constrained-join-proxy-18>.

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

[DNSSD-SRP] Lemon, T. and S. Cheshire, "Service Registration Protocol for DNS-Based Service Discovery", RFC 9665, DOI 10.17487/RFC9665, June 2025, <https://www.rfc-editor.org/rfc/rfc9665>.

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

[GRASP] Bormann, C., Carpenter, B., Ed., and B. Liu, Ed., "GeneRic Autonomic Signaling Protocol (GRASP)", RFC 8990, DOI 10.17487/RFC8990, May 2021, <https://www.rfc-editor.org/rfc/rfc8990>.

[JWS-VOUCHER] Werner, T. and M. Richardson, "JWS signed Voucher Artifacts for Bootstrapping Protocols", Work in Progress, Internet-Draft, draft-ietf-anima-jws-voucher-16, 15 January 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-anima-jws-voucher-16>.

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

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

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <https://www.rfc-editor.org/rfc/rfc2782>.

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

[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/rfc/rfc5280>.

[RFC7390] Rahman, A., Ed. and E. Dijk, Ed., "Group Communication for the Constrained Application Protocol (CoAP)", RFC 7390, DOI 10.17487/RFC7390, October 2014, <https://www.rfc-editor.org/rfc/rfc7390>.

[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/rfc/rfc8174>.

[RFC8368] Eckert, T., Ed. and M. Behringer, "Using an Autonomic Control Plane for Stable Connectivity of Network Operations, Administration, and Maintenance (OAM)", RFC 8368, DOI 10.17487/RFC8368, May 2018, <https://www.rfc-editor.org/rfc/rfc8368>.

[RFC9148] van der Stok, P., Kampanakis, P., Richardson, M., and S. Raza, "EST-coaps: Enrollment over Secure Transport with the Secure Constrained Application Protocol", RFC 9148, DOI 10.17487/RFC9148, April 2022, <https://www.rfc-editor.org/rfc/rfc9148>.

[RFC9176] Ams端ss, C., Ed., Shelby, Z., Koster, M., Bormann, C., and P. van der Stok, "Constrained RESTful Environments (CoRE) Resource Directory", RFC 9176, DOI 10.17487/RFC9176, April 2022, <https://www.rfc-editor.org/rfc/rfc9176>.

[RFC9483] Brockhaus, H., von Oheimb, D., and S. Fries, "Lightweight Certificate Management Protocol (CMP) Profile", RFC 9483, DOI 10.17487/RFC9483, November 2023, <https://www.rfc-editor.org/rfc/rfc9483>.

8.2. Informative References

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

[HRW98] Thaler, D. D. and C. V. Ravishankar, "Using Name-Based Mappings to Increase Hit Rates", 1998, <https://www.microsoft.com/en-us/research/wp-content/uploads/2017/02/HRW98.pdf>.

[I-D.eckert-anima-grasp-dnssd]
Eckert, T. T., Boucadair, M., Jacquenet, C., and M. H. Behringer, "DNS-SD Compatible Service Discovery in GeneRic Autonomic Signaling Protocol (GRASP)", Work in Progress, Internet-Draft, draft-eckert-anima-grasp-dnssd-08, 7 July 2025, <https://datatracker.ietf.org/doc/html/draft-eckert-anima-grasp-dnssd-08>.

[I-D.ietf-bess-evpn-fast-df-recovery]
Brissette, P., Sajassi, A., Burdet, L. A., Drake, J., and J. Rabadan, "Fast Recovery for EVPN Designated Forwarder Election", Work in Progress, Internet-Draft, draft-ietf-bess-evpn-fast-df-recovery-12, 20 November 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-bess-evpn-fast-df-recovery-12>.

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

[RFC8894] Gutmann, P., "Simple Certificate Enrolment Protocol", RFC 8894, DOI 10.17487/RFC8894, September 2020, <https://www.rfc-editor.org/rfc/rfc8894>.

Contributors

Thomas Werner
Siemens AG
Germany
Email: thomas-werner@siemens.com
URI: https://www.siemens.com/

Steffen Fries
Siemens AG
Germany
Email: steffen.fries@siemens.com
URI: https://www.siemens.com/

Hendrik Brockhaus
Siemens AG
Germany
Email: hendrik.brockhaus@siemens.com
URI: https://www.siemens.com/

Michael Richardson
Canada
Phone: +41 44 878 9200
Email: mcr+ietf@sandelman.org

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

Authors' Addresses

Toerless Eckert (editor)
Futurewei USA
United States of America
Email: tte@cs.fau.de

Esko Dijk
IoTconsultancy.nl
Email: esko.dijk@iotconsultancy.nl