Workgroup: Network Working Group
Internet-Draft:
draft-hallambaker-mesh-callsign
Published: 23 October 2022
Intended Status: Informational
Expires: 26 April 2023
Authors: P. M. Hallam-Baker
ThresholdSecrets.com

Mathematical Mesh 3.0 Part VII: Mesh Callsign Service

Abstract

The Mesh Callsign Registry is a name registry that provides a mapping from human-friendly callsigns to root of trusts and a service assigned by the callsign holder to service the account bound to that callsign. An append only sequence authenticated by means of a Merkle Tree and periodic third party notarizations provides ground truth for the integrity of the registry and for all the assertions enrolled in the sequence.

https://mailarchive.ietf.org/arch/browse/mathmesh/Discussion of this draft should take place on the MathMesh mailing list (mathmesh@ietf.org), which is archived at .

Status of This Memo

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

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

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

This Internet-Draft will expire on 26 April 2023.

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction

The Mesh Callsign registry performs three principal functions:

- *Provides ground truth for notarized records.
- *Provides a quasi-permanent mapping from human-friendly callsigns to root of trusts.
- *Provides a means of discovering the service currently servicing an account bound to a root of trust.

For example, Alice registers the callsign @alice, binding the callsign to her Mesh account profile and her current Mesh service provider (@provisional). The callsign may now be used (subject to the relevant authorization criteria) to contact Alice through any Mesh messaging protocol or through any of the Internet protocols specified in the contact declaration(s) Alice has published to that account.

Unlike traditional Internet account addresses, callsign registrations are the property of the holder, do not require any maintenance fees or rent and cannot be reassigned without the consent of the holder except in exceptional circumstances according to a well-defined dispute resolution process.

Mesh accounts are likewise the property of the holder and not the service provider servicing them. Should Alice be dissatisfied with the service provided by @provisional, she can change to another provider at any time.

As with the URIs that support the World Wide Web, Mesh callsigns are conceptually distinct from the Mesh itself. While the primary purpose of the callsign registry is to provide for Mesh 'account portability' similar to the 'number portability' required of telephone providers, the infrastructure established is general purpose and may be applied to obtain roots of trust for use with IPSEC, DNSEC, TLS, etc.

1.1. Wafer-Thin Registry Model

Providing a permanent name assignment on the basis of a one-off fee requires close attention to be paid to the allocation of costs within the system. An architecture that requires a registry to perform ongoing services with respect to a registration cannot do so on the basis of a one-time fee.

In the DNS naming infrastructure overseen by ICANN, a distinction is made between a Registry responsible for the administration of a 'top level domain' and the Registrars responsible for servicing domain

name holders as customers. While this 'thin registry' model offloads the cost of customer service to the registrars, the Registry is responsible for the difficult and expensive task of providing the authoritative DNS service for the zone.

In the wafer-thin registry model, the task of providing the callsign query function is performed by the registrars and others from the append only logs published by the registry. This has the effect of assigning only the fixed one-time registration costs to the registry and placing the variable costs on the registrar (Figure xxx).

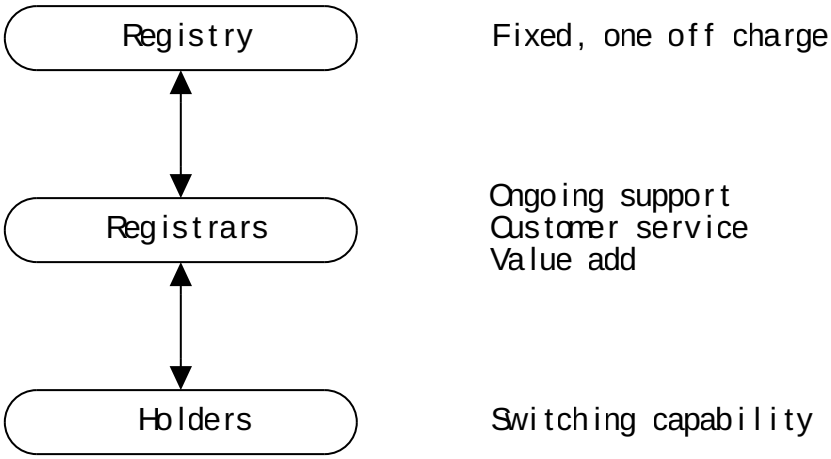

Figure 1: Callsign Registry Principals and Communication Graph.

In the DNS thin registry model, the registry is a single point of failure and a successful denial of service attack against the registry would result in loss of service for every Internet user. Consequently, such registries must be over-built and over-provisioned to ensure such attacks cannot possibly succeed. In the wafer-thin registry model, the query service is distributed across multiple registrars, each of which services a specific community. The single point of failure is eliminated and the responsibility for query service placed with parties that are much better placed to distinguish legitimate query traffic from abuse.

1.2. Transparency

The technical implementation of the callsign registry is designed to provide transparency, that is the ability for any party to audit the actions of any other party using only publicly available information.

The callsign registry is realized an append only sequence authenticated by means of a Merkle Tree. Each callsign registration

or update is made by means of a signed assertion appended to the end of the sequence.

The callsign registry periodically issues a signed witness token asserting that the apex value of the Merkle Tree has a particular value and incorporates witness tokens issued by others in its own log as a notarization event. Over time, these interactions between the Registry, the Registrars and their customers establishes a Mesh of Trus such that it is impossible for any party to defect undetected unless every other party colludes in the deception. For example, consider the following circumstance in which Alice, Bob, their providers and the registry perform the following series of cross notarization events:

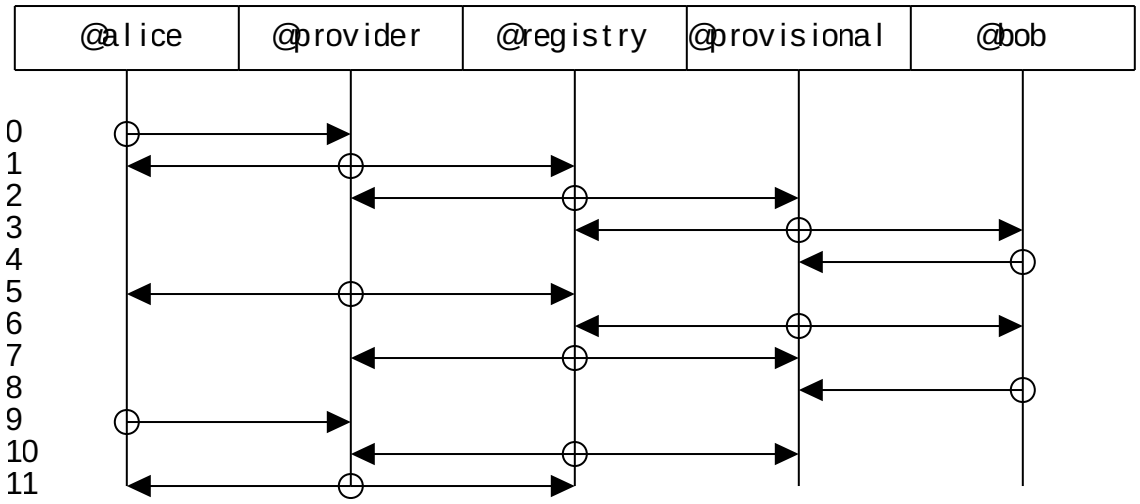

Figure 2: Mesh of Trust generated through Cross-Notarization.

At the start of the process, the only party with proof of the integrity of Bob's sequence is Bob. Creation of a witness token that is enrolled in the provider @provisional increases the support structure by one, etc.

0 @bob
1 @bob & @provisional
2 @bob & @provisional
3 @bob & @provisional & @registry & @provider
4 @bob & @provisional & @registry & @provider
5 @bob & @provisional & @registry & @provider & @alice

Figure 3: Support structure for integrity of Bob's sequence at time 0.

After a short series of interactions, the trust mesh is complete and each of the participants can validate Bob's sequence according to themselves as the ultimate root of trust. Thus Alice can validate

Bob's claim that a document was created by a certain date relying on herself as the root of trust:

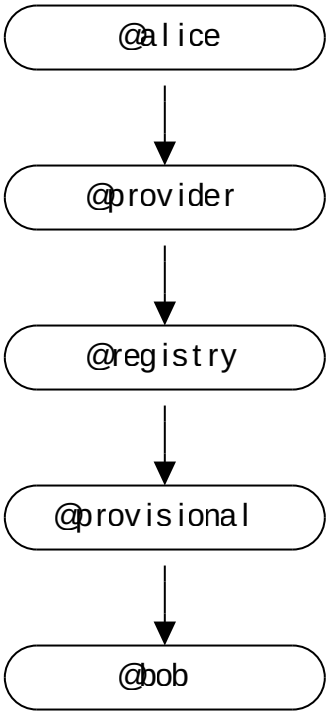

Figure 4: Notarization Trust Relationships.

1.3. Dispute Resolution

Except in exceptional circumstances, callsign registrations cannot be transferred without the express permission of the holder as evidenced by a digital assertion signed under a key authorized by the callsign registration.

One such exceptional circumstance is the case in which an intellectual property claim is made against the use of the callsign. A set of principles for resolving such disputes and a dispute resolution process for deciding them is therefore required. Such questions are outside the scope of this document, the provision of technical infrastructure to support them is not.

While there are many jurisdictions that might assert sovereignty with respect to intellectual property claims, it is in practice only necessary to consider those capable of enforcing their decisions. Fortunately, this very fact has resulted in international treaties such as the Madrid protocol.

1.4. Name resolution

The WebPKI provides a means of authenticating organizations by reducing the validation criteria to the determination that a specific process was used to validate certificate requests. This approach has been successful largely because organizations and in particular corporations are entities that come into being by the specific act of registration by a government entity.

Natural persons do not come into being through government action. Consequently, attempts to apply this approach to identification of natural persons have met with limited success at best. The problem of establishing a PKI for identification of natural persons has long been recognized as one of the hardest challenges in the security field.

Callsigns provide a solution to this by establishing a name space in which the registration of a name is coextensive with the registration of the root of trust in which the name is to be interpreted. Since this is a completely new name space with no pre-existing commitments, we are free to design the namespace to meet the needs of the PKI rather than having to adapt the PKI to the vagaries of the namespace.

While introducing a new namespace solves the problem of establishing a PKI for natural persons, it leaves us with the problem of how to make use of the new infrastructure on devices and protocols that only understand the legacy namespaces. While Alice does not respond to HTTP requests, she owns many devices that do and these should be accessible through the trust and naming contexts established by Alice's callsign registration.

These requirements are addressed by mapping the names established by the callsign registry to an unused portion of DNS. Devices bound to Alice's callsign '@alice' being accessible through the DNS pseudo-domain alice.mesh.

For example, is Alice buys a thermostat and binds it to her Mesh profile, the device is assigned the sub-callsign hvac@alice and the DNS name hvac.alice.mesh. The device may then be provisioned with all the credentials and discovery services required to provide Alice with seamless thermostat service within her home:

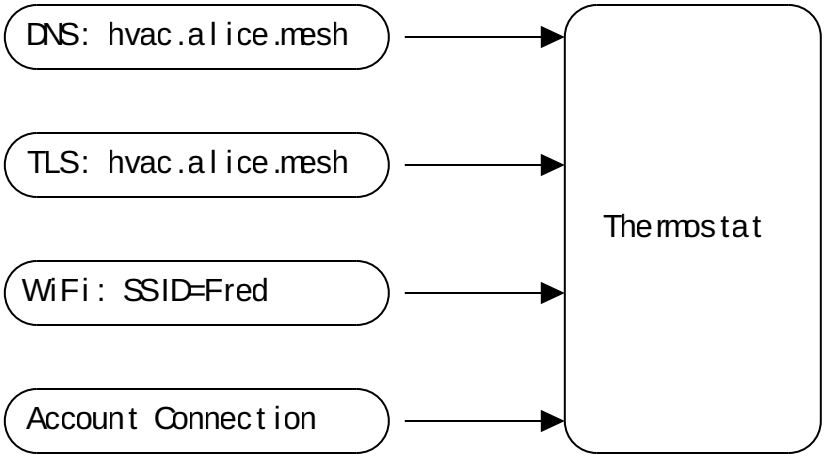

Figure 5: Thermostat connected to Alice's Mesh.

1.5. Trusted Third Parties

The Mesh Trust architecture is based on the principle that every person has the right to be their own root of trust. This principle is realized in the application of cross notarization through which the ultimate authority for Alice on entries in Bob's personal catalogs is Alice herself.

But trust management is a difficult and time-consuming task and just as it likely that Alice outsources the trusted task of guarding her money to her bank, it is likely that Alice will delegate at least some trust management decisions to trusted third parties of her choosing.

In the WebPKI, Trusted Third Parties (CAs) enable Internet commerce by establishing accountability. The fact that a Web site is owned by a registered business in a particular jurisdiction does not in itself guarantee that the Web site owner is trustworthy. But the fact that they are registered in that jurisdiction means that they are accountable to it should they breach the terms of a contract or attempt to perpetrate some fraud.

Trusted Third Parties provide additional information to parties relying on a callsign through accreditations. An accreditation is a trust assertion that states that the holder of a callsign has been determined to meet some criteria according to a specific set of practices meeting a given policy. For example:

*The holder is a registered business.

*The holder is approved by some business accreditation service.

*The holder is the registered holder of a specific trademark.

The ability to bind registered trademarks to a callsign holder is of particular interest when Mesh Messaging is being used for a transaction requiring a particular degree of trust.

For example, Alice might opt to use her Internet connected watch as a second factor authentication device connected to her brokerage account. When Alice attempts to perform a high value transaction purchasing a large number of penny stocks, she receives a confirmation message on her watch. The accredited trademark of the brokerage is used to authenticate the message to Alice before she reads and responds to it.

2. Definitions

This section presents the related specifications and standards....

2.1. Related Specifications

The Mesh Callsign registry is a component part of the Mathematical Mesh [draft-hallambaker-mesh-architecture] and makes use of the data formats and service formats described therein. In particular:

Uniform Data Fingerprint [draft-hallambaker-mesh-udf]. Describes the UDF format used to represent cryptographic nonces, keys and content digests in the Mesh and the use of Encrypted Authenticated Resource Locators (EARLs) and Strong Internet Names (SINs) that build on the UDF platform.

Data at Rest Encryption [draft-hallambaker-mesh-dare]. Describes the cryptographic message and append-only sequence formats used in Mesh applications and the Mesh Service protocol.

JSON-BCD Encoding [draft-hallambaker-jsonbcd]. Describes extensions to the JSON serialization format to allow direct encoding of binary data (JSON-B), compressed encoding (JSON-C) and extended binary data encoding (JSON-D). Each of these encodings is a superset of the previous one so that JSON-B is a superset of JSON, JSON-C is a superset of JSON-B and JSON-D is a superset of JSON-C.

2.2. Defined Terms

This document makes use of the terms defined in [draft-hallambaker-mesh-architecture].

2.3. Requirements Language

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2.4. Implementation Status

The implementation status of the reference code base is described in the companion document [draft-hallambaker-mesh-developer].

The examples in this document were created on 18-Oct-22 1:16:01 PM. Out of 249 examples, 52 failed.

2.5. Reserved Callsigns

The following callsigns are reserved identifiers in the callsign registry. When used in this document, these callsigns refer to the following parties:

@alice, @bob, @carol, @doug The generic end users, Alice, Bob, Carol and Doug.

@callsign, @callsign1, @callsign2 Generic callsigns

@corporation, @customer, @competitor A generic corporation and its customer and competitor.

@eve An eavesdropper

@grace A government representative

@heidi A malicious designer for cryptographic standards

@judy A judge who may be called upon to resolve a potential dispute between participants.

@mallet A malicious party engaged in an active attack

@provider, @provisional Mesh Service Providers

@quartermaster The callsign of the registry quartermaster.

@registry The callsign registry profider.

@sybil, @sybil0, @sybil-1, @sybil-n A pseudonymous attacker, who usually uses a large number of identities.

@ted A trusted arbitrator, who acts as a neutral third party.

@wendy A whistleblower, who is an insider with privileged access capable of divulging information.

@Firstname_Lastname A generic user whose name is 'Firstname Lastname'

3. Callsign Syntax

A canonical Callsign consists of one or more canonical Unicode characters that are specified in a single character page as defined by a Registry character page registration.

A presentation form of a callsign consists of the symbol '@' followed by a string of one or more Unicode characters which has the canonical form of the callsign as defined by a Registry character page registration.

This approach responds to lessons learned from the experience of internationalized DNS names, in particular the threat of a homograph attack in which similar or identical looking characters from different character sets are combined to create a domain name that looks like a name that has already been registered.

Specifications for two character pages are proposed in this document:

CharacterPageDigits The digits [0-9] are canonical.

The spacing characters ' ' and '_' are non-canonical and map to the empty string.

CharacterPageLatin Includes the CharacterPageDigits page by reference

The characters [a-z] are canonical.

The characters [A-Z] and accented Latin characters are non-canonical and map to the characters [a-z] according to usage.

Each callsign has a single, unique canonical form that is determined by replacing each non-canonical character with the replacement specified in the character page registration.

For example, Alice may register her callsign as @alice, @Alice, @Alic?, etc. All of which have the same canonical form 'alice'.

Bob can send a message to Alice by entering any callsign presentation that maps to the same canonical form as the callsign registered by Alice. But when Bob receives a message from Alice, her callsign **SHOULD** be presented in the registered presentation if this is known.

Character page definitions are published through the registry. This allows new code pages to be added at any time without the need to update existing clients (unless the client lacks the fonts required to present the callsign).

Character page definitions **MAY** be extended by adding additional variants. A variant consists of a mapping of a single variant character to a group of one or more canonical characters. A variant character mapping **MUST NOT** specify any character that is canonical in that code page as its source.

It is of course inevitable that the process of canonicalization will result in two or more words that have distinct and different meanings mapping to the same callsign just as it is inevitable that any technological infrastructure that attempts to span diverse cultural practices arising from centuries of use will not be able to satisfy everyone.

3.1. Reserved Characters

The following characters are reserved and **MUST NOT** be used in a callsign

@ Reserved to prevent confusion and to enable use in transitive naming.

. Reserved to enable interoperability with DNS names.

: Reserved to avoid ambiguity when used in URIs

Reserved to avoid ambiguity with URI fragment identifier syntax.

% Reserved for future use to allow URI percent-encoding of callsigns to be specified.

3.2. Additional character pages

Additional character pages will be added after a proof-of-concept deployment has been achieved for the Latin character page. Character pages to be considered include:

Emoji character page Why not? Will map to a specific set of symbols via some escape sequence.

Arabic, Chinese, Cyrillic, Devanagari, Hebrew, etc. Just need to have a domain expert specify how to do these.

3.3. Callsign Delegation

The form delegate@callsign **MAY** be used to create a subordinate callsign.

*Provide names for IoT devices

*Roles within an organization

For example, Alice might assign the callsign hvac@alice to the thermostat controlling her home heating and ventilation system.

If Alice works for @corporation, she might be assigned the subordinate callsign alice@corporation for the duration of her employment. This allows for a smooth transfer of responsibilities when Alice eventually leaves. It is clear to anyone who interacts with Alice through the callsign alice@corporation, that the 'end' of the communication and the trust relationship in this case is @corporation and not Alice.

In circumstances in which Mesh callsigns are used in combinaton with RFC822 email addresses, the DNS pseudo domain .mesh **MAY** be used for disambiguation. Thus @alice **MAY** be written as @alice.mesh and alice@corporation **MAY** be written as alice@corporation.mesh.

3.4. Transitive Callsign

The form callsign1@@callsign2 **MAY** be used to specify the party that the party holding callsign2 knows as callsign1.

This format **MAY** be used to support SPKI/SDSI forms of transitive naming. This may be of use in constructing identifiers to refer to elements of Notarization proof chains. 'registry@@provider' being an identifier for a Witness token generated by @registry and enrolled by @provider.

3.5. Mapping to DNS Names

The DNS pseudo domain .mesh is used to provide a reserved space to which Mesh callsigns **MAY** be mapped through assertions specified in Registry entries. The .mesh domain is conceptually distinct from those currently offered by ICANN. Instead of the zone being maintained as a DNS zone delegated from the DNS root, the zone entries are determined from the entries in the callsign registry:

For example: To resolve the domain hvac.alice.mesh, a Mesh aware application requests callsign resolution on @alice.

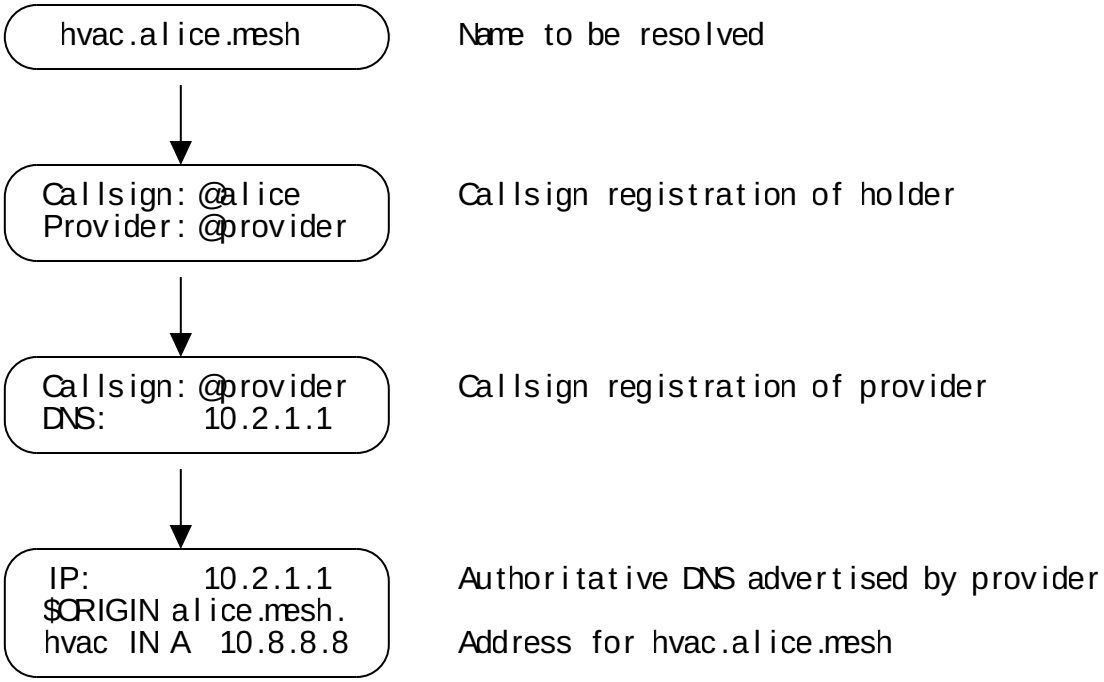

Figure 6: DNS resolution of hvac.alice.mesh.

The callsign entry for @alice does not specify a DNS field but it does specify that the callsign is serviced by @provider. The callsign entry for @provider specifies a DNS entry for an authoritative name server for all the DNS zones being serviced by the provider. This includes service for the zone alice.mesh which contains an IP address for hvac.alice.mesh.

A provider of DNS resolution service **MAY** provide delegation service for the complete .mesh domain by determining the corresponding delegation entries from the callsign registry as they are entered:

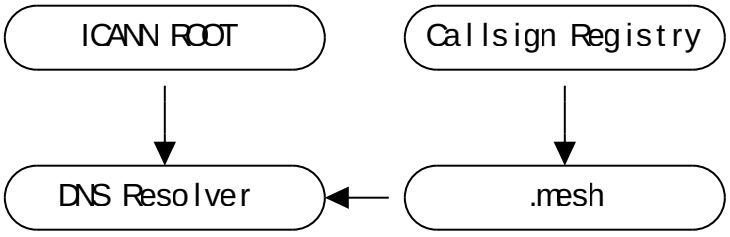

Figure 7: DNS resolution of Callsign Pseudo-Domain.

3.5.1. DNS record validations

Delegations in the DNS pseudo domain .mesh are authenticated by means of the signatures on the Mesh callsign registrations and notarization of the registry sequence through witness entries. While this does not provide a DNSSEC validation path, the path provided does not require the constant maintenance of resigning the zone required in DNSSEC.

The delegated domains (e.g. alice.mesh) **MAY** be DNSSEC signed. The signing key for the domain being authenticated by means of a security policy.

4. Registry Sequence

The Registry Append Only Sequence is a DARE Sequence of type Merkle Tree. The following entry types are defined:

Registration Registration of a callsign, accreditation or challenge.

Page Specification of a character page specifying characters allowed in a callsign.

Notarization Entry of a signed witness token from another sequence (e.g. a sequence maintained by a registrar).

4.1. Callsign Assignment

Callsigns are registered by means of a Registration entry containing an enveloped callsign signed by the callsign holder.

The callsign entry contains all the information necessary to audit future requests to update or transfer the callsign registration.

4.1.1. Aliases

A callsign registration **MAY** contain an alias entry directing resolution requests to another callsign.

One very important difference between callsigns and DNS names is that a callsign resolver has access to the entire callsign registry. Consequently, it is not necessary to place limits on the length of path resolution for aliases etc. since these can be calculated during processing of the sequence. Detection of indirection loops is still required of course.

4.1.2. Initial Registration

Initial registration records have the reason field 'Initial'.

Alice registers the callsign alice it has the canonical form alice.


```
{
  "Callsign":{
    "Id":"alice",
    "Presentation":"alice",
    "Holder":"MBHS-EKNA-7EJZ-VMUQ-HI5U-4FH3-RV3T",
    "Service":"provisional"}}
```


The registrar receives the request and forwards it to the registry which creates a registration record that is enrolled in the callsign log:


```
{
  "Registration":{
    "Id":"NAVQ-OJVT-PRML-SR5S-BUH6-QAUM-UPTM",
    "Entry":[{
        "dig":"S512"},
      "ewogICJDYWxsc2lnbiI6IHsKICAgICJJZCI6ICJhbGljZSIsCiAgICAiUH
  Jlc2VudGF0aW9uIjogImFsaWNlIiwKICAgICJIb2xkZXIiOiAiTUJIUy1FS05BLTd
  FSlotVk1VUS1ISTVVLTRGSDMtUlYzVCIsCiAgICAiU2VydmljZSI6ICJwcm92aXNp
  b25hbCJ9fQ",
      {
        "signatures":[{
            "alg":"S512",
            "kid":"MBHS-EKNA-7EJZ-VMUQ-HI5U-4FH3-RV3T",
            "signature":"EeTMZx7MAVskcP87n7bj3yz9ukgGs22etareDWrf
  jBSmtcbv3p6zYuamKlVrgO3LdHpC7TDNLcuAEBc2OMj2ycTj_blGkAX9sw1RuUREy
  M4S3HV0g-PUwSzDFkBItBKMpE0nVKq35d8SehWdA9CSCDMA"}
          ],
        "PayloadDigest":"IKtQPyuBxPdM81GeFEMCsTK1WG4xyp_xsI3yUjyu
  URfIIAGgYVh0JVd5M59V742SHthSC5abV30gCu3W4q6_FA"}
      ],
    "Submitted":"2021-04-03T14:47:49Z",
    "Reason":"Initial"}}
```


4.1.3. Update

A callsign holder **MAY** update their callsign registration at any time.

- To change the presentation form of their callsign.

- To specify an alias to which attempts at resolving the callsign should be mapped.

*To change their service provider.

*To make changes to their Profile or DNS entries.

Callsign updates **MUST NOT** change the canonical form of a callsign. Such changes are by definition a registration of a different callsign.

Alice decides to change her service provider to provider. She also decides that her callsign looks better with an initial capital and so changes the presentation form to Alice. This change does not require require a new callsign to be registered as this has tghe same canonical form alice.


```
{
  "Callsign":{
    "Id":"alice",
    "Presentation":"Alice",
    "Holder":"MBHS-EKNA-7EJZ-VMUQ-HI5U-4FH3-RV3T",
    "Service":"provider"}}
```


The registrar receives the request and forwards it to the registry which creates a registration record that is enrolled in the callsign log:


```
{
  "Registration":{
    "Id":"NBBT-SEMX-O32L-D7M3-IHP6-GJO4-XSED",
    "Entry":[{
        "dig":"S512"},
      "ewogICJDYWxsc2lnbiI6IHsKICAgICJJZCI6ICJhbGljZSIsCiAgICAiUH
  Jlc2VudGF0aW9uIjogIkFsaWNlIiwKICAgICJIb2xkZXIiOiAiTUJIUy1FS05BLTd
  FSlotVk1VUS1ISTVVLTRGSDMtUlYzVCIsCiAgICAiU2VydmljZSI6ICJwcm92aWRl
  ciJ9fQ",
      {
        "signatures":[{
            "alg":"S512",
            "kid":"MBHS-EKNA-7EJZ-VMUQ-HI5U-4FH3-RV3T",
            "signature":"l5kZXYUpcoieHn8uIIRLnF4aP3uyi7Ekk6bpWkeg
  7NWLwcImhXPDrXJo4HRPMWFExJtPpv41gPcAcAlgTIe3RISgI5_GNcuIaBJdwGytp
  1dbl0HsiciPvxDKaTC1RWUPA1wG_UXzC4Xc-gO64ZK4BzQA"}
          ],
        "PayloadDigest":"Et8xDoEm1MDB4X3o9vAUiDWvbjlZNVR5ZsQMQeX6
  vX8OCcEzL5lkD9zDxPwc8ecug_CCxdNpMoVhOeecygZ2Wg"}
      ],
    "Submitted":"2021-04-03T14:47:49Z",
    "Reason":"Initial"}}
```

4.1.4. Voluntary Transfer

A callsign holder **MAY** transfer their holdership of the callsign to another party in the same mannaer as making an update but specifying the Reason 'Transfer' instead of 'Update'. This tells relying parties that ownership of the callsign has been transferred and that messages sent to that callsign will be received by a different party.

4.1.5. Administrative Transfer

The reason 'administrative' is used to inform relying parties that the callsign was transferred without the permission of the callsign holder and that the signature on the callsign record is that of the administrative entity that authorized the transfer.

The administrative criteria and processes under which administrative transfers occur are outside the scope of this document. Such processes **MAY** require registration of a Challenge assertion to put the callsign holder on notice that their holdership has been challenged. Mesh Service Providers acting on behalf of their users **SHOULD** alert callsign holders when their callsign is challenged.

4.2. Character Page Description

Character page descriptions specify a set of characters which **MAY** be used within a following callsign registration. The Page record specifies the range of characters that are canonical and the mapping of variant characters to canonical form.

For example, the CharacterPageDigits Page specifies ten canonical characters, the digits, 0,1,2,...,9 and two variant characters which **MAY** be used to space letters when presenting a callsign. Both variants map to the empty string meaning that they are simply discarded when compiling the canonical form.

```
{
  "Page":{
    "Id":"CharacterPageDigits",
    "CharacterSpans":[{
        "Canonical":{
          "First":48,
          "Last":57}},
      {
        "MapString":{
          "First":32,
          "Target":""}},
      {
        "MapString":{
          "First":95,
          "Target":""}}
      ]}}
```

Page records **MAY** incorporate other Page records by reference provided that they have been defined earlier in the sequence.

4.3. Notarization

The registry produces DARE Witness tokens at periodic intervals as determined by policy. For example, policy might dictate that a Witness token be produced every day or every hour or that the interval between witness token generation be determined by the rate at which entries are added to the sequence.

Registrars **MAY** also produce DARE Witness tokens and request that they be enrolled in the log as notarization entries. A notarization entry is not signed directly but is instead authenticated by the Merkle Tree authenticating the sequence as a whole.

The first time that the Mesh service provider provisional requests Notarization by the Registry, it creates a witness token on its local log:

```
{
  "Witness":{
    "Id":"provisional",
    "Issuer":"provisional",
    "Apex":"z4PhNX7vuL3xVChQ1m2AB9Yg5AULVxXcg_SpIdNs6c5H0NE8XYXys
  P-DGNKHfuwvY7kxvUdBeoGlODJ6-SfaPg"}}
```

The witness value is signed by provisional to create a Notarization of the witness value:

```
{
  "Notarization":{
    "Entries":[[{
          "dig":"S512"},
        "ewogICJXaXRuZXNzIjogewogICAgIklkIjogInByb3Zpc2lvbmFsIiwK
  ICAgICJJc3N1ZXIiOiAicHJvdmlzaW9uYWwiLAogICAgIkFwZXgiOiAiejRQaE5YN
  3Z1TDN4VkNoUTFtMkFCOVlnNUFVTFZ4WGNnX1NwSWROczZjNUgwTkU4WFlYeXNQCi
  AgLURHTktIZnV3dlk3a3h2VWRCZW9HbE9ESjYtU2ZhUGcifX0",
        {
          "signatures":[{
              "alg":"S512",
              "kid":"MAG2-SFGU-YIE3-YY56-CFQQ-XE52-XOWA",
              "signature":"uGmMvEKz73JsGPP91MktmumFeOgx_plUqtvn5P
  4Xf396XVuBRl7gdmJsdOq4_UUFo8jM222XNcUA4HlzN0OZU9jcOKi7svesib__jnM
  2mjuiLrx2dQImPKquw7_VLOlxeNsfwttixS3iRKbZQrpswxkA"}
            ],
          "PayloadDigest":"J9_gKXKo_Q7JWwc2IvoNt23Sb6oF3HWYpCQosA
  WflDcCSrsfMxIsSNmpkQmIpRsFGoItTK-5TDhXIEV3MTCN6w"}
        ]
      ]}}
```


The Notarization value is sent to the registry which enrolls the Notarization in its local log, creates a Witness value on the entry containing the provider Notarization, and returns its own Notarization token containing the Witness value it has most recently created.

The next time that the service provider requests Notarization, it creates a Witness token as before and includes a Proof path to the previous token:

```
{
  "Notarization":{
    "Entries":[[{
          "dig":"S512"},
        "ewogICJXaXRuZXNzIjogewogICAgIklkIjogInByb3Zpc2lvbmFsIiwK
  ICAgICJJc3N1ZXIiOiAicHJvdmlzaW9uYWwiLAogICAgIkFwZXgiOiAiejRQaE5YN
  3Z1TDN4VkNoUTFtMkFCOVlnNUFVTFZ4WGNnX1NwSWROczZjNUgwTkU4WFlYeXNQCi
  AgLURHTktIZnV3dlk3a3h2VWRCZW9HbE9ESjYtU2ZhUGcifX0",
        {
          "signatures":[{
              "alg":"S512",
              "kid":"MAG2-SFGU-YIE3-YY56-CFQQ-XE52-XOWA",
              "signature":"uGmMvEKz73JsGPP91MktmumFeOgx_plUqtvn5P
  4Xf396XVuBRl7gdmJsdOq4_UUFo8jM222XNcUA4HlzN0OZU9jcOKi7svesib__jnM
  2mjuiLrx2dQImPKquw7_VLOlxeNsfwttixS3iRKbZQrpswxkA"}
            ],
          "PayloadDigest":"J9_gKXKo_Q7JWwc2IvoNt23Sb6oF3HWYpCQosA
  WflDcCSrsfMxIsSNmpkQmIpRsFGoItTK-5TDhXIEV3MTCN6w"}
        ]
      ],
    "Proof":{}}}
```


[NB, generation and verification of proof is not currently supported in the reference code]

This time, the Registry verifies the proof path before entering the token.

4.4. Accreditation

Issue and use of third party accreditations is outside the scope of this document.

5. Callsign Registration Interaction

The CallSign registry operates as a standard Mesh Account with the callsign @registry.

The registry periodically fetches callsign registration requests and processes them to produce a valid log.

Callsigns are registered and updated by sending a request message containing a valid signed CallsignBinding with the necessary proof of payment (if required).

6. Callsign Resolution

The callsign resolution protocol supports the Hello and Query transactions.

Request specifies the Callsign

Response returns the binding (if it exists) and a statement specifying how current the callsign provider's data is.

Alice adds an Internet connected Thermostat to her home. The thermostat has a built in Web server to allow the settings to be set by a Web browser. During the onboarding process, the thermostat is assigned a domain name {Policies.ThermostatDns}

Alice's Mesh Service Provider maintains an authoritative DNS server for Alice's Callsign DNS zone {Policies.AliceDns}. The IP addresses of this server are specified in the callsign registration of her Mesh Service Provider:

```
{
  "Callsign":{
    "Id":"provider",
    "Presentation":"provider",
    "Holder":"MC33-EPHG-GILF-YB7P-RQF4-DPWZ-TRYY",
    "Service":"@provider",
    "Dns":["10.2.1.1",
      "10.2.1.2",
      "2001:db8::1001"
      ]}}
```

The authoritative DNS server publishes a link from which a Mesh DNS Profile specifying the security policy for the zone MAY be obtained:

[This is probably a prefixed TXT record of some sort.]

The security policy for the zone states that the DNS zone is signed using DNSSEC and the thermostat supports TLS/1.2 transport layer security:

```
{
  "ProfileDns":{
    "SecurityPolicies":[{
        "CName":["alice.mesh"
          ],
        "Protocol":["dns"
          ],
        "Enhancements":["dnssec"
          ],
        "Roots":[{
            "Udf":"MBHS-EKNA-7EJZ-VMUQ-HI5U-4FH3-RV3T",
            "PublicParameters":{
              "PublicKeyECDH":{
                "crv":"Ed448",
                "Public":"p0H30Oh6fm2HxxcYTqhWNJGLUHD7SQgBHjtRUIE
  zdCGDzAo9ytlAgDwQ_0UT6uF3jBWSnqmJtxMA"}}}
          ]},
      {
        "CName":["hvac.alice.mesh"
          ],
        "Protocol":["http",
          "https"
          ],
        "Enhancements":["tls1.2"
          ],
        "Require":true,
        "Roots":[{
            "Udf":"MBHS-EKNA-7EJZ-VMUQ-HI5U-4FH3-RV3T",
            "PublicParameters":{
              "PublicKeyECDH":{
                "crv":"Ed448",
                "Public":"p0H30Oh6fm2HxxcYTqhWNJGLUHD7SQgBHjtRUIE
  zdCGDzAo9ytlAgDwQ_0UT6uF3jBWSnqmJtxMA"}}}
          ]}
      ]}}
```

A non Mesh-aware browser can access the Web site and establish a TLS connection to the thermostat provided that 1) the browser uses a DNS service that supports use of Mesh callsign zones and 2) the device provides a certificate set that allows the browser to build a valid path to a root it trusts.

A Mesh aware browser can access the Web site directly and enforce the security policy directly. Thus preventing a downgrade attack to an insecure site.

7. Schemas

7.1. Callsign log entries

7.1.1. Profile and account registration

7.1.2. Structure: ProfileRegistry

Describes a callsign registry.

[No fields]

7.1.3. Structure: ProfileResolver

Describes a callsign resolver.

RegistryAddress: String (Optional) The address of the registry

EnvelopedProfileRegistry: Enveloped (Optional) The registry that this resolver resolves.

CommonEncryption: KeyData (Optional) Key currently used to encrypt data under this profile

CommonAuthentication: KeyData (Optional) Key used to authenticate requests made under this user account. This key SHOULD NOT be provisioned to any device except for the purpose of enabling account recovery.

7.1.4. Callsign registration and transfer

The following classes are used as common elements in Mesh profile specifications.

7.1.5. Structure: Registration

A callsign or Notarization registration

Id: String (Optional) Unique registration identifier

Entry: Enveloped (Optional) The signed callsign binding

Submitted: DateTime (Optional) The UTC time instant that the claim was submitted.

Registrar: String (Optional) Callsign of the registrar that made the registration request

PriorId: String (Optional) If present, specifies a previous registration with the same identifier.

Reason: String (Optional)
Reason for creating a registration: Initial/ Update/ Voluntary/ Administrative/ Revoke

7.1.6. Structure: CatalogedRegistration

Canonical: String (Optional) The canonical form of the callsign.

Id: String (Optional) Unique registration identifier

EnvelopedRegistration: Enveloped (Optional) The registration entry for the item.

7.1.7. Character page definition

7.1.8. Structure: Page

Id: String (Optional) Character page identifier

Allow: String [0..Many] Additional allowed pages.

7.1.9. Structure: CharacterSpan

First: Integer (Optional) The first character in the range (inclusive)

Last: Integer (Optional) The last character in the range (inclusive), if ommitted or equal to zero, this is the same as Last.

7.1.10. Structure: Canonical

Inherits: CharacterSpan
Canonical character span.

[No fields]

7.1.11. Structure: MapChar

Inherits: CharacterSpan
Specifies a variant mapping the range of characters First..First+n to a range of characters Target..Target+n. Where n = Last - First+1

Target: Integer (Optional) The character that First is mapped to.

7.1.12. Structure: MapString

Inherits: CharacterSpan

Specifies a mapping of non canonical characters in the range specified by First..Last to the string Target.

Target: String (Optional) Specifies a character string that the Source character(s) are mapped to. If count is greater than 1, all the characters map to the same string.

7.1.13. Notarization

7.1.14. Structure: Notarization

Entries: Enveloped [0..Many] Enveloped witness value of a specific append only log.

Proof: Proof (Optional) Proof path validating the previous notary token that was entered in the log.

7.1.15. Trust Assertions

7.1.16. Structure: Challenge

Registers a challenge to one or more callsigns that have been registered.

Subjects: String [0..Many] The callsigns subject to challenge

Basis: String [0..Many] The basis for the challenge

7.2. Message Classes

7.2.1. Structure: CallsignRegistrationRequest

Connection request message. This message contains the information

EnvelopedCallsignBinding: Enveloped (Optional) The enveloped binnding of the callsign to the profile.

7.2.2. Structure: CallsignRegistrationResponse

Registered: Boolean (Optional) True if and only if a new registration was created.

CatalogedRegistration: CatalogedRegistration (Optional) The resulting catalog entry if accepted or the prior registration otherwise.

Reason: String (Optional) Reason for refusing the registration.

7.2.3. Structure: ProcessResultCallsignRegistration

CallsignRegistrationResponse: CallsignRegistrationResponse

(Optional)

7.3. Registry service

7.3.1. Registry Account

7.3.2. Structure: CatalogedRegistry

EnvelopedConnectionAddress: Enveloped (Optional) The connection allowing control of the registry.

EnvelopedProfileRegistry: Enveloped (Optional) The Mesh profile

EnvelopedActivationCommon: Enveloped (Optional) The activation data for the registry.

7.3.3. Structure: ActivationApplicationRegistry

AccountEncryption: KeyData (Optional) Key used to decrypt registry messages.

AdministratorSignature: KeyData (Optional) Key or capability used to sign the registry log

EnvelopedConnectionService: Enveloped (Optional) Signed connection service delegation allowing the device to access the account.

7.3.4. Structure: ApplicationEntryRegistry

EnvelopedActivation: Enveloped (Optional)

7.3.5. Shared Classes

HTTP Well Known Service Prefix: /.well-known/callsignreg

7.3.6. Message: CallsignRequest

Base class for all requests made to a registrar

[No fields]

7.3.7. Message: CallsignResponse

Base class for all response messages. Contains only the status code and status description fields.

[No fields]

7.3.8. Transactions

7.4. Transaction: Register

Request: RegisterRequest
Response: RegisterResponse Request registration of a callsign.

7.4.1. Message: RegisterRequest

Request registration of a callsign.

Inherits: CallsignRequest
PriorId: String (Optional) If present, specifies a previous registration with the same identifier.

Reason: String (Optional) Reason for creating a registration: Initial/ Update/ Voluntary/ Administrative/ Revoke

EnvelopedCallsign: Enveloped (Optional) The callsign request signed by the party to which the callsign is to be granted.

7.4.2. Message: RegisterResponse

Inherits: CallsignResponse Reports the success or failure of a registration.

Result: Enveloped (Optional) The registration (if created).

7.5. Transaction: Notarize

Request: NotarizeRequest
Response: NotarizeResponse Request inclusion of a signed witness value in the chain.

7.5.1. Message: NotarizeRequest

Request registration of a callsign.

Inherits: CallsignRequest The notary token to be included.
Witness: Enveloped (Optional)

7.5.2. Message: NotarizeResponse

Inherits: CallsignResponse Reports the success or failure of a registration.

Witness: Enveloped (Optional) The most recent witness value created by the service.

Notarization: Enveloped (Optional) A notarization entry that includes the witness value specified in the request.

7.6. Registrar service

7.6.1. Shared Classes

HTTP Well Known Service Prefix: /.well-known/tbs1

7.6.2. Transactions

7.7. Transaction: Query

Request: QueryRequest
Response: QueryResponse Request resolution of a profile bound to a callsign or registration identifier.
Returns an envelope containing the signed registration (if found).

7.7.1. Message: QueryRequest

Request resolution of a profile bound to a callsign or registration identifier.

CallSign: String (Optional) The callsign being requested in canonical form.

RegistrationId: String (Optional) The registration identifier of a registration in the log.

LogId: String (Optional) The unique identifier of an append only log whose signed Notarization entry is requested.

7.7.2. Message: QueryResponse

Return the result of a QueryRequest

Result: Enveloped (Optional) The registration specified in the result (if found).

Notarization: Enveloped (Optional) The latest notarization entry corresponding to the specified log.

8. Security Considerations

8.1. Names

8.1.1. Impersonation

8.1.2. Homograph attack

8.1.3. Malicious Intellectual Property Claim

8.2. Credential Loss

8.2.1. Loss

8.2.2. Disclosure

8.3. Breach of Faith

8.3.1. Registrar

8.3.2. Registry

8.4. Quantum Cryptanalysis

9. IANA Considerations

This document requires no IANA actions.

10. Acknowledgements

11. Appendix A: Latin Character Page

```
{
  "Page":{
    "Id":"CharacterPageLatin",
    "Allow":["CharacterPageDigits"
      ],
    "CharacterSpans":[{
        "Canonical":{
          "First":97,
          "Last":122}},
      {
        "MapChar":{
          "First":65,
          "Last":90,
          "Target":97}},
      {
        "MapString":{
          "First":192,
          "Target":"a"}},
      {
        "MapString":{
          "First":193,
          "Target":"a"}},
      {
        "MapString":{
          "First":194,
          "Target":"a"}},
      {
        "MapString":{
          "First":195,
          "Target":"a"}},
      {
        "MapString":{
          "First":196,
          "Target":"ae"}},
      {
        "MapString":{
          "First":197,
          "Target":"aa"}},
      {
        "MapString":{
          "First":198,
          "Target":"ae"}},
      {
        "MapString":{
          "First":199,
          "Target":"c"}},
      {
        "MapString":{
          "First":200,
```

```
    "Target":"e"}},
{
  "MapString":{
    "First":201,
    "Target":"e"}},
{
  "MapString":{
    "First":202,
    "Target":"e"}},
{
  "MapString":{
    "First":203,
    "Target":"e"}},
{
  "MapString":{
    "First":204,
    "Target":"i"}},
{
  "MapString":{
    "First":205,
    "Target":"i"}},
{
  "MapString":{
    "First":206,
    "Target":"i"}},
{
  "MapString":{
    "First":207,
    "Target":"i"}},
{
  "MapString":{
    "First":208,
    "Target":"d"}},
{
  "MapString":{
    "First":209,
    "Target":"n"}},
{
  "MapString":{
    "First":210,
    "Target":"o"}},
{
  "MapString":{
    "First":211,
    "Target":"o"}},
{
  "MapString":{
    "First":212,
    "Target":"o"}},
```

```
{
  "MapString":{
    "First":213,
    "Target":"o"}},
{
  "MapString":{
    "First":214,
    "Target":"oe"}},
{
  "MapString":{
    "First":215,
    "Target":"x"}},
{
  "MapString":{
    "First":217,
    "Target":"u"}},
{
  "MapString":{
    "First":218,
    "Target":"u"}},
{
  "MapString":{
    "First":219,
    "Target":"u"}},
{
  "MapString":{
    "First":220,
    "Target":"u"}},
{
  "MapString":{
    "First":221,
    "Target":"y"}},
{
  "MapString":{
    "First":223,
    "Target":"ss"}},
{
  "MapString":{
    "First":224,
    "Target":"a"}},
{
  "MapString":{
    "First":225,
    "Target":"a"}},
{
  "MapString":{
    "First":226,
    "Target":"a"}},
{
```

```
  "MapString":{
    "First":227,
    "Target":"a"}},
{
  "MapString":{
    "First":228,
    "Target":"ae"}},
{
  "MapString":{
    "First":229,
    "Target":"aa"}},
{
  "MapString":{
    "First":230,
    "Target":"ae"}},
{
  "MapString":{
    "First":231,
    "Target":"c"}},
{
  "MapString":{
    "First":232,
    "Target":"e"}},
{
  "MapString":{
    "First":233,
    "Target":"e"}},
{
  "MapString":{
    "First":234,
    "Target":"e"}},
{
  "MapString":{
    "First":235,
    "Target":"e"}},
{
  "MapString":{
    "First":236,
    "Target":"i"}},
{
  "MapString":{
    "First":237,
    "Target":"i"}},
{
  "MapString":{
    "First":238,
    "Target":"i"}},
{
  "MapString":{
```

```
    "First":239,
    "Target":"i"}},
{
  "MapString":{
    "First":240,
    "Target":"o"}},
{
  "MapString":{
    "First":241,
    "Target":"n"}},
{
  "MapString":{
    "First":242,
    "Target":"o"}},
{
  "MapString":{
    "First":243,
    "Target":"o"}},
{
  "MapString":{
    "First":244,
    "Target":"o"}},
{
  "MapString":{
    "First":245,
    "Target":"o"}},
{
  "MapString":{
    "First":246,
    "Target":"oe"}},
{
  "MapString":{
    "First":247,
    "Target":"-"}},
{
  "MapString":{
    "First":249,
    "Target":"u"}},
{
  "MapString":{
    "First":250,
    "Target":"u"}},
{
  "MapString":{
    "First":251,
    "Target":"u"}},
{
  "MapString":{
    "First":252,
```

```
        "Target":"u"}},
    {
      "MapString":{
        "First":253,
        "Target":"y"}}
    ]}}
```

12. Normative References

[draft-hallambaker-jsonbcd]
Hallam-Baker, P., "Binary Encodings for JavaScript Object Notation: JSON-B, JSON-C, JSON-D", Work in Progress, Internet-Draft, draft-hallambaker-jsonbcd-23, 23 October 2022, <https://datatracker.ietf.org/doc/html/draft-hallambaker-jsonbcd-23>.

[draft-hallambaker-mesh-architecture]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part I: Architecture Guide", Work in Progress, Internet-Draft, draft-hallambaker-mesh-architecture-21, 23 October 2022, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-architecture-21>.

[draft-hallambaker-mesh-dare]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part III : Data At Rest Encryption (DARE)", Work in Progress, Internet-Draft, draft-hallambaker-mesh-dare-16, 23 October 2022, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-dare-16>.

[draft-hallambaker-mesh-udf]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part II: Uniform Data Fingerprint.", Work in Progress, Internet-Draft, draft-hallambaker-mesh-udf-17, 23 October 2022, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-udf-17>.

[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>.

13. Informative References

[draft-hallambaker-mesh-developer]
Hallam-Baker, P., "Mathematical Mesh: Reference Implementation", Work in Progress, Internet-Draft, draft-hallambaker-mesh-developer-10, 27 July 2020, <https://

datatracker.ietf.org/doc/html/draft-hallambaker-mesh-developer-10>.