Workgroup: Network Working Group
Internet-Draft:
draft-hallambaker-mesh-protocol
Published: 27 July 2020
Intended Status: Informational
Expires: 28 January 2021
Authors: P. M. Hallam-Baker
ThresholdSecrets.com

Mathematical Mesh 3.0 Part V: Protocol Reference

Abstract

The Mathematical Mesh 'The Mesh' is an end-to-end secure infrastructure that facilitates the exchange of configuration and credential data between multiple user devices. The core protocols of the Mesh are described with examples of common use cases and reference data.

[Note to Readers]

Discussion of this draft takes place on the MATHMESH mailing list (mathmesh@ietf.org), which is archived at https://mailarchive.ietf.org/arch/search/?email_list=mathmesh.

This document is also available online at http://mathmesh.com/Documents/draft-hallambaker-mesh-protocol.html.

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 28 January 2021.

Copyright Notice

Table of Contents

1. Introduction

This document describes the Mesh Service protocol supported by Mesh Services, an account-based protocol that facilitates exchange of data between devices connected to a Mesh profile and between Mesh accounts.

Mesh Service Accounts support the following services:

*Provides the master persistence store for the Catalogs and Spools associated with the account.

*Enables synchronization of Catalogs and Spools with connected devices.

*Enforces access control on inbound Mesh Messages from other users and other Mesh Services.

*Authenticates outbound Mesh Messages, certifying that they comply with abuse mitigation policies.

A Mesh Profile **MAY** be bound to multiple Mesh Service Accounts at the same time but only one Mesh Service Account is considered to be authoritative at a time. Users may add or remove Mesh Service Accounts and change the account designated as authoritative at any time.

The Mesh Services are build from a very small set of primitives which provide a surprisingly extensive set of capabilities. These primitives are:

Hello
: Describes the features and options provided by the service and provides a 'null' transaction which **MAY** be used to establish an authentication ticket without performing any action,

CreateAccount, DeleteAccount Manage the creation and deletion of accounts at the service.

Status, Download, Upload Support synchronization of Mesh containers between the service (Master) and the connected devices (Replicas).

Connect Initiate the process of connecting a device to a Mesh profile from the device itself.

Post Request that a Mesh Message be transferred to one or more Mesh Accounts.

Although these functions could in principle be used to replace many if not most existing Internet application protocols, the principal value of any communication protocol lies in the size of the audience it allows them to communicate with. Thus, while the Mesh Messaging service is designed to support efficient and reliable transfer of messages ranging in size from a few bytes to multiple terabytes, the near-term applications of these services will be to applications that are not adequately supported by existing protocols if at all.

2. Definitions

This section presents the related specifications and standard, the terms that are used as terms of art within the documents and the terms used as requirements language.

2.1. 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 [RFC2119].

2.2. Defined Terms

The terms of art used in this document are described in the *Mesh Architecture Guide* [draft-hallambaker-mesh-architecture].

2.3. Related Specifications

The architecture of the Mathematical Mesh is described in the *Mesh Architecture Guide* [draft-hallambaker-mesh-architecture]. The Mesh documentation set and related specifications are described in this document.

2.4. Implementation Status

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

3. Mesh Service

A Mesh Service is a minimally trusted service. In particular a user does not need to trust a Mesh service to protect the confidentiality or integrity of most data stored in the account catalogs and spools.

Unless the use of the Mesh Service is highly restricted, a user does need to trust the Mesh Service in certain respects:

Data Loss A service could refuse to respond to requests to download data.

Integrity (Stale Data) The use of Merkle Trees limits but does not eliminate the ability of a Mesh Service to respond to requests with stale data.

Messaging A service could reject requests to post messages to or accept messages from other mesh users.

This risk is a necessary consequence of the fact that the Mesh Service Provider is accountable to other Mesh Service Providers for abuse originating from their service.

Traffic analysis A Mesh Service has knowledge of the number of Mesh Messages being sent and received by its users and the addresses to which they are being sent to or received from.

The need to trust the Mesh Service in these respects is mitigated by accountability and the user's ability to change Mesh Service providers at any time they choose with minimal inconvenience.

It is possible that some of these risks will be reduced in future versions of the Mesh Service Protocol but it is highly unlikely that these can be eliminated entirely without compromising practicality or efficiency.

3.1. Data Model

The design of the Mesh Service model followed a quasi-formal approach in which the system was reduced to schemas which could in principle be rendered in a formal development method but without construction of proofs.

Like the contents of Mesh Accounts, a Mesh Service may be represented by a collection of catalogs and spools, for example:

Account Catalog
Contains the account entries.

Incident Spool Reports of potential abuse

Backup of the service **MAY** be implemented using the same container synchronization mechanism used to synchronize account catalogs and spools.

3.2. Partitioning

Mesh Services supporting a large number of accounts or large activity volume **MAY** partition the account catalog between one or more hosts using the usual tiered service model in which a front-end server receives traffic for any account hosted at the server and routes the request to the back-end service that provides the persistence store for that account.

In addition, the Mesh Service Protocol supports a 'direct connection' partitioning model in which devices are given a DNS name which **MAY** allow for direct connection to the persistence host or to a front-end service offering service that is in some way specific to that account.

4. Protocol Bindings

Mesh Service transactions are mapped to an underlying messaging and transport protocol. The following binding

Mesh Services **MUST** support the Web Service binding specified in this document and **MAY** support the UDP binding currently in development.

4.1. DNS Web Service Discovery

The DNS Web Service discovery mechanism is used to discover Mesh Services regardless of the protocol binding .The service name, DNS prefix and and .well-known service suffix are specified as follows:

*Service Name: mmm

*DNS Prefix: _mmm._tcp

*Well Known service suffix: /.well-known/mmm

4.2. Web Service Protocol Binding

The Web Service Protocol binding makes use of the most widely deployed and used protocols:

*Discovery: DNS Service discovery

*Transport: TLS

*Application: HTTP

*Presentation: DARE Message

*Encoding: JSON, JSON-B

The chief limitations of the Web Service Protocol Binding are that the use of TCP based transport results in unsatisfactory latency for some applications and that the HTTP application layer only serves to allow a host to support multiple services on the same TCP/IP port.

4.2.1. Transport Security

Mesh Services **MUST** offer TLS transport and **MAY** offer non TLS transport. MESH clients **SHOULD** use TLS transport when connecting to a MESH service.

TLS version 1.3 [RFC8446] or higher **MUST** be supported. Client authentication **SHOULD NOT** be used.

4.2.2. HTTP Message Binding

All messages are exchanged as HTTP POST transactions. Support for and use of HTTP/1.1 [RFC7230] is **REQUIRED**. Services **MAY** support HTTP/2.

In contrast to other approaches to the design of Web Services, the only use made of the HTTP transport is to distinguish between different services on the same host using the Host header and .well-known convention and for message framing. No use is made of the URI request line to identify commands, nor are the caching or proxy capabilities of HTTP made use of.

4.2.3. Request

The HTTP request **MAY** contain any valid HTTP header specified in [RFC7230].

Request Line URI /well-known/<service> (unless overridden using a TXT path attribute)

Request Line Method POST

Host: Header <domain>

Content-Encoding As specified in section yy below.

Content-Type As specified in section zz below.

Content-Length or Transfer-Encoding
As specified in [RFC7230].

Payload The content payload as specified in section XX below.

[Note, this is showing the payload, not the binding as is intended because the current code doesn't implement it as intended yet]

```
{
  "Hello": {}}
```

4.2.4. Response

The response **MAY** contain any HTTP response header but since JWB services do not make use of HTTP caching and messages are not intended to be modified by HTTP intermediaries, only a limited number of headers have significance:

Response Code The HTTP response code. This is processed as described in section zz below.

Content-Type As specified in section zz below.

Content-Length or Transfer-Encoding As specified in [RFC7230].

Cache-Control Since the only valid HTTP method for a JWB request is POST, JWB responses are not cacheable. The use of the cache-control header is therefore unnecessary. However, experience suggests that reviewers find it easier to understand protocol specifications if they are reminded of the fact that caching is neither supported nor desired.

[Note, this is showing the payload, not the binding as is intended because the current code doesn't implement it as intended yet]

```
{
  "MeshHelloResponse": {
    "Status": 201,
    "Version": {
      "Major": 3,
      "Minor": 0,
      "Encodings": [{
          "ID": ["application/json"]}]},
    "EnvelopedProfileService": [{
        "dig": "SHA2"},
      "ewogICJQcm9maWxlU2VydmljZSI6IHsKICAgICJLZXlPZmZsaW5lU2l
  nbmF0dXJlIjogewogICAgICAiVURGIjogIk1BWFotSVBHNy1WSEFPLVdRVVQtQ
  0JUTC1ITFMyLVNLT1ciLAogICAgICAiUHVibGljUGFyYW1ldGVycyI6IHsKICA
  gICAgICAiUHVibGljS2V5RUNESCI6IHsKICAgICAgICAgICJjcnYiOiAiRWQ0N
  DgiLAogICAgICAgICAgIlB1YmxpYyI6ICJiY2paaUQzQmdQcThOSEE4VFFzbEp
  RbzBzOE14R1o0VWRPUV8zcGZBWGhmUkdLdDh3aVhlCiAgZTl6MzZ4V0JxVGVKR
  khhd09uTWNPcmlBIn19fSwKICAgICJLZXlFbmNyeXB0aW9uIjogewogICAgICA
  iVURGIjogIk1EM0gtM09LRi1LTDM0LU02Q0YtSkVHVS1BMkVNLUNUWVUiLAogI
  CAgICAiUHVibGljUGFyYW1ldGVycyI6IHsKICAgICAgICAiUHVibGljS2V5RUN
  ESCI6IHsKICAgICAgICAgICJjcnYiOiAiWDQ0OCIsCiAgICAgICAgICAiUHVib
  GljIjogInBJcjlQeTVrWm52QU9paTZ0REdRejc1ZFVCNTNpQkIxM0ZKQXZzSDJ
  NdHlJU2RQWFZ3TWMKICBrU3FOTGl5TzBpUTJaanE5QkRXVmpmOEEifX19fX0",
      {
        "signatures": [{
            "alg": "SHA2",
            "kid": "MAXZ-IPG7-VHAO-WQUT-CBTL-HLS2-SKOW",
            "signature": "Q8tU8qUnPvpxEqpZd_TWkvIN8_1YtRCkxGqSk2Exad_
                4PlWzr
  AzG_cX9VT5UmOaJ6cmi-lM8hF8Am-OgRNkxGkb1y_OYtaKBXnUeBGHG3jtnplU
  vx3VlWReJn0VqYMiTfcAQ5PKNujHbf4iCso6PSAcA"}],
        "PayloadDigest": "tWH90O5uV-b6VvIjiwiClRkJ6D2C64GZLXqZk5pAhxt
            QW
  wqcCzKM3f7s6slOuJ6fZ267uDg6QQ2UF55_vjsiSA"}],
    "EnvelopedProfileHost": [{
        "dig": "SHA2"},
      "ewogICJQcm9maWxlSG9zdCI6IHsKICAgICJLZXlPZmZsaW5lU2lnbmF
  0dXJlIjogewogICAgICAiVURGIjogIk1CM1otSzVITC1FS0FCLTZNNUotM1BLW
  S1CSVlCLUpZSzUiLAogICAgICAiUHVibGljUGFyYW1ldGVycyI6IHsKICAgICA
  gICAiUHVibGljS2V5RUNESCI6IHsKICAgICAgICAgICJjcnYiOiAiRWQ0NDgiL
  AogICAgICAgICAgIlB1YmxpYyI6ICJnamlvV3pVYllqNkxMQ2I0NGRrYUw4STV
  wOTFpcE5IaHJOTVF0VlFJbFA2aXg3MmZVWnFYCiAgRFpBTmdKSzNpOXU4RGNIT
  XhpV0hsalFBIn19fSwKICAgICJLZXlBdXRoZW50aWNhdGlvbiI6IHsKICAgICA
  gIlVERiI6ICJNQ1g3LTZQQ0QtV1BISC1WRjRMLVZRNVgtSVNJSC1UVk0zIiwKI
  CAgICAgICAgIlB1YmxpY1BhcmFtZXRlcnMiOiB7CiAgICAgICAgICAgICAgIlB1YmxpY0tleUV
  DREgiOiB7CiAgICAgICAgICAgICAgICAiY3J2IjogIlg0NDgiLAogICAgICAgICAgICAgIlB1Y
  mxpYyI6ICJPbGViSVA0Y3otOEVWV05lRFBVTE9rajJlczd2dTdBUF9OTEo1MWs
  xSDBYb1N0UWtFaWtQCiAgQ19vM3FCVmJsVUh2Uk0wblBFQmxjdGdBIn19fX19",
      {
        "signatures": [{
```

```
            "alg": "SHA2",
            "kid": "MB3Z-K5HL-EKAB-6M5J-3PKY-BIYB-JYK5",
            "signature": "Ten3iq7kUJdif_NE94cr-RJ0OJD8qrG-9gj7eu-M6ae
                oqLk3_
  Ew9MqJ8nxvpI13YWEq1Zb9uRfaAbcOghmncfeN9rwTeoApfGEE5lZ5O2h97s1m
  inEKmUEWMOoT2-CBkVSZlH50rYADhbJ9N1ZpWRDAA"}],
        "PayloadDigest": "2K1y20hWCmxcFb_DzTRpHlEyPbqySb-TPF8cA6Y_T5i
            fc
  OupwDkIwlLqWQBML9IxCZyqtZ37vJh4DkAA6LGWWg"}]}}
```


4.3. DARE Message Encapsulation

The payload of the HTTP requests and responses is a DARE Message whose payload contains the Mesh Service request or response.

The DARE Message encapsulation is used to authenticate the request or response data. The form of the authentication depending on the credentials available to the sender at the time the request is made.

Mesh Service **MUST** support the use of Mutually Authenticated Key Exchange [draft-hallambaker-mesh-security] to establish the Master Key used for authentication of requests and responses.

Requests and Responses **MUST** be authenticated. Requests and Responses **MUST** be encrypted if the transport is not encrypted and **MAY** be encrypted otherwise.

4.3.1. Null Authentication

Null Authentication **MAY** be used to make a Hello Request.

The Null Authentication mechanism **MUST NOT** be used for any Mesh Service request or response other than a Hello request.

Since the Mutually Authenticated key exchange requires both parties to know the public key of the other, it is not possible for a client to authenticate itself to the service until it has obtained the service public key. One means by which the client **MAY** obtain the service public key is by requesting the service return the credential in a Hello transaction.

4.3.2. Device Authentication

Device Authentication is used in two circumstances

- *When requesting creation of an account
- *When a device is requesting connection to a profile.

4.3.3. Profile Authentication

Profile Authentication has the same form as Device Authentication except that the client provides its Device Connection Assertion as part of the request:

4.3.4. Ticket Authentication

Ticket Authentication is used after a device has obtained an authentication ticket from a service. The ticket is returned in the response to a previous Profile Authentication exchange.

4.4. Payload Encoding

The Dare Message payload of a Hello request **MUST** be encoded in JSON encoding. The payload of all other requests **MUST** be in either JSON encoding or one of the encodings advertised as being accepted in a Hello response from the Service. Services **MUST** accept JSON encoding and **MAY** support the JSON-B or JSON-C encodings as specified in this document. Services **MUST** generate a response that is compatible with the DARE Message Content-Type specified in the request.

JSON was originally developed to provide a serialization format for the JavaScript programming language [ECMA-262]. While this approach is generally applicable to the type systems of scripting programming languages, it is less well matched to the richer type systems of modern object oriented programming languages such as Java and C#.

Working within a subset of the capabilities of JSON allows a Web Service protocol to be accessed with equal ease from either platform type. The following capabilities of JSON are avoided:

The ability to use arbitrary strings as field names.

The use of JSON objects to define maps directly

The following data field types are used:

Integer Integer values are encoded as JSON number values.

String Test strings are encoded as JSON text strings.

Boolean Boolean values are encoded as JSON 'false', 'true' or 'null' tokens according to value.

Sequence Sequences of data items that are encoded as JSON arrays

Object of known type Objects whose type is known to the receiver are encoded as JSON objects

Object of variable type
Objects whose type is not known to the receiver are encoded as JSON objects containing a single field whose name describes the type of the object value and whose value contains the value.

Binary Data Byte sequences are converted to BASE64-url encoding [RFC4648] and encoded as JSON string values.

Date Time Date Time values are converted to Internet time format as described in [RFC3339] and encoded as JSON string values.

4.5. Error handling and response codes

It is possible for an error to occur at any of the three layers in the Web Service binding:

Service Layer

HTTP Layer

Transport Layer

Services **SHOULD** always attempt to return error codes at the highest level possible. However, it is clearly impossible for a connection that is refused at the Transport layer to return an error code at the HTTP layer. It is however possible for a HTTP layer error response to contain a content body.

In the case that a response contains both a HTTP response code and a well-formed payload containing a response, the payload response **SHALL** have precedence.

5. Service Description

The Hello transaction is used to determine the features supported by the service and obtain the service credentials

The request payload:

```
{
  "Hello": {}}
```

The response payload:

```
{
  "MeshHelloResponse": {
    "Status": 201,
    "Version": {
      "Major": 3,
      "Minor": 0,
      "Encodings": [{
          "ID": ["application/json"]}]},
    "EnvelopedProfileService": [{
        "dig": "SHA2"},
      "ewogICJQcm9maWxlU2VydmljZSI6IHsKICAgICJLZXlPZmZsaW5lU2l
  nbmF0dXJlIjogewogICAgICAiVURGIjogIk1BWFotSVBHNy1WSEFPLVdRVVQtQ
  0JUTC1ITFMyLVNLT1ciLAogICAgICAiUHVibGljUGFyYW1ldGVycyI6IHsKICA
  gICAgICAiUHVibGljS2V5RUNESCI6IHsKICAgICAgICAgICJjcnYiOiAiRWQ0N
  DgiLAogICAgICAgICAgIlB1YmxpYyI6ICJiY2paaUQzQmdQcThOSEE4VFFzbEp
  RbzBzOE14R1o0VWRPUV8zcGZBWGhmUkdLdDh3aVhlCiAgZTl6MzZ4V0JxVGVKR
  khhd09uTWNPcmlBIn19fSwKICAgICJLZXlFbmNyeXB0aW9uIjogewogICAgICA
  iVURGIjogIk1EM0gtM09LRi1LTDM0LU02Q0YtSkVHVS1BMkVNLUNUWVUiLAogI
  CAgICAiUHVibGljUGFyYW1ldGVycyI6IHsKICAgICAgICAiUHVibGljS2V5RUN
  ESCI6IHsKICAgICAgICAgICJjcnYiOiAiWDQ0OCIsCiAgICAgICAgICAiUHVib
  GljIjogInBJcjlQeTVrWm52QU9paTZ0REdRejc1ZFVCNTNpQkIxM0ZKQXZzSDJ
  NdHlJU2RQWFZ3TWMKICBrU3FOTGl5TzBpUTJaanE5QkRXVmpmOEEifX19fX0",
      {
        "signatures": [{
            "alg": "SHA2",
            "kid": "MAXZ-IPG7-VHAO-WQUT-CBTL-HLS2-SKOW",
            "signature": "Q8tU8qUnPvpxEqpZd_TWkvIN8_1YtRCkxGqSk2Exad_
                4PlWzr
  AzG_cX9VT5UmOaJ6cmi-lM8hF8Am-OgRNkxGkb1y_OYtaKBXnUeBGHG3jtnplU
  vx3VlWReJn0VqYMiTfcAQ5PKNujHbf4iCso6PSAcA"}],
        "PayloadDigest": "tWH90O5uV-b6VvIjiwiClRkJ6D2C64GZLXqZk5pAhxt
            QW
  wqcCzKM3f7s6slOuJ6fZ267uDg6QQ2UF55_vjsiSA"}],
    "EnvelopedProfileHost": [{
        "dig": "SHA2"},
      "ewogICJQcm9maWxlSG9zdCI6IHsKICAgICJLZXlPZmZsaW5lU2lnbmF
  0dXJlIjogewogICAgICAiVURGIjogIk1CM1otSzVITC1FS0FCLTZNNUotM1BLW
  S1CSVlCLUpZSzUiLAogICAgICAiUHVibGljUGFyYW1ldGVycyI6IHsKICAgICA
  gICAiUHVibGljS2V5RUNESCI6IHsKICAgICAgICAgICJjcnYiOiAiRWQ0NDgiL
  AogICAgICAgICAgIlB1YmxpYyI6ICJnamlvV3pVYllqNkxMQ2I0NGRrYUw4STV
  wOTFpcE5IaHJOTVF0VlFJbFA2aXg3MmZVWnFYCiAgRFpBTmdKSzNpOXU4RGNIT
  XhpV0hsalFBIn19fSwKICAgICJLZXlBdXRoZW50aWNhdGlvbiI6IHsKICAgICA
  gIlVERiI6ICJNQ1g3LTZQQ0QtV1BISC1WRjRMLVZRNVgtSVNJSC1UVk0zIiwKI
  CAgICAgICAgIlB1YmxpY1BhcmFtZXRlcnMiOiB7CiAgICAgICAgICAgICAgIlB1YmxpY0tleUV
  DREgiOiB7CiAgICAgICAgICAgICAgICAiY3J2IjogIlg0NDgiLAogICAgICAgICAgICAgIlB1Y
  mxpYyI6ICJPbGViSVA0Y3otOEVWV05lRFBVTE9rajJlczd2dTdBUF9OTEo1MWs
  xSDBYb1N0UWtFaWtQCiAgQ19vM3FCVmJsVUh2Uk0wblBFQmxjdGdBIn19fX19",
      {
        "signatures": [{
```

```
            "alg": "SHA2",
            "kid": "MB3Z-K5HL-EKAB-6M5J-3PKY-BIYB-JYK5",
            "signature": "Ten3iq7kUJdif_NE94cr-RJ0OJD8qrG-9gj7eu-M6ae
                oqLk3_
  Ew9MqJ8nxvpI13YWEq1Zb9uRfaAbcOghmncfeN9rwTeoApfGEE5lZ5O2h97s1m
  inEKmUEWMOoT2-CBkVSZlH50rYADhbJ9N1ZpWRDAA"}],
        "PayloadDigest": "2K1y20hWCmxcFb_DzTRpHlEyPbqySb-TPF8cA6Y_T5i
            fc
  OupwDkIwlLqWQBML9IxCZyqtZ37vJh4DkAA6LGWWg"}]}}
```

6. Account Management

A Mesh Account is bound to a Mesh Service by completing a CreateAccount transaction with the service.

The client requesting the account creation specifies the ProfileMesh profile describing the requested account and lists of initial entries to populate the devices and contacts catalogs. Additional catalogs **MAY** be synchronized if the account creation request is accepted.

The request payload:

```
{
  "Hello": {}}
```

The response payload:

```
{
  "MeshHelloResponse": {
    "Status": 201,
    "Version": {
      "Major": 3,
      "Minor": 0,
      "Encodings": [{
          "ID": ["application/json"]}]},
    "EnvelopedProfileService": [{
        "dig": "SHA2"},
      "ewogICJQcm9maWxlU2VydmljZSI6IHsKICAgICJLZXlPZmZsaW5lU2l
  nbmF0dXJlIjogewogICAgICAiVURGIjogIk1BWFotSVBHNy1WSEFPLVdRVVQtQ
  0JUTC1ITFMyLVNLT1ciLAogICAgICAiUHVibGljUGFyYW1ldGVycyI6IHsKICA
  gICAgICAiUHVibGljS2V5RUNESCI6IHsKICAgICAgICAgICJjcnYiOiAiRWQ0N
  DgiLAogICAgICAgICAgIlB1YmxpYyI6ICJiY2paaUQzQmdQcThOSEE4VFFzbEp
  RbzBzOE14R1o0VWRPUV8zcGZBWGhmUkdLdDh3aVhlCiAgZTl6MzZ4V0JxVGVKR
  khhd09uTWNPcmlBIn19fSwKICAgICJLZXlFbmNyeXB0aW9uIjogewogICAgICA
  iVURGIjogIk1EM0gtM09LRi1LTDM0LU02Q0YtSkVHVS1BMkVNLUNUWVUiLAogI
  CAgICAiUHVibGljUGFyYW1ldGVycyI6IHsKICAgICAgICAiUHVibGljS2V5RUN
  ESCI6IHsKICAgICAgICAgICJjcnYiOiAiWDQ0OCIsCiAgICAgICAgICAiUHVib
  GljIjogInBJcjlQeTVrWm52QU9paTZ0REdRejc1ZFVCNTNpQkIxM0ZKQXZzSDJ
  NdHlJU2RQWFZ3TWMKICBrU3FOTGl5TzBpUTJaanE5QkRXVmpmOEEifX19fX0",
      {
        "signatures": [{
            "alg": "SHA2",
            "kid": "MAXZ-IPG7-VHAO-WQUT-CBTL-HLS2-SKOW",
            "signature": "Q8tU8qUnPvpxEqpZd_TWkvIN8_1YtRCkxGqSk2Exad_
                4PlWzr
  AzG_cX9VT5UmOaJ6cmi-lM8hF8Am-OgRNkxGkb1y_OYtaKBXnUeBGHG3jtnplU
  vx3VlWReJn0VqYMiTfcAQ5PKNujHbf4iCso6PSAcA"}],
        "PayloadDigest": "tWH90O5uV-b6VvIjiwiClRkJ6D2C64GZLXqZk5pAhxt
            QW
  wqcCzKM3f7s6slOuJ6fZ267uDg6QQ2UF55_vjsiSA"}],
    "EnvelopedProfileHost": [{
        "dig": "SHA2"},
      "ewogICJQcm9maWxlSG9zdCI6IHsKICAgICJLZXlPZmZsaW5lU2lnbmF
  0dXJlIjogewogICAgICAiVURGIjogIk1CM1otSzVITC1FS0FCLTZNNUotM1BLW
  S1CSVlCLUpZSzUiLAogICAgICAiUHVibGljUGFyYW1ldGVycyI6IHsKICAgICA
  gICAiUHVibGljS2V5RUNESCI6IHsKICAgICAgICAgICJjcnYiOiAiRWQ0NDgiL
  AogICAgICAgICAgIlB1YmxpYyI6ICJnamlvV3pVYllqNkxMQ2I0NGRrYUw4STV
  wOTFpcE5IaHJOTVF0VlFJbFA2aXg3MmZVWnFYCiAgRFpBTmdKSzNpOXU4RGNIT
  XhpV0hsalFBIn19fSwKICAgICJLZXlBdXRoZW50aWNhdGlvbiI6IHsKICAgICA
  gIlVERiI6ICJNQ1g3LTZQQ0QtV1BISC1WRjRMLVZRNVgtSVNJSC1UVk0zIiwKI
  CAgICAgICAgIlB1YmxpY1BhcmFtZXRlcnMiOiB7CiAgICAgICAgICAgICAgIlB1YmxpY0tleUV
  DREgiOiB7CiAgICAgICAgICAgICAgICAiY3J2IjogIlg0NDgiLAogICAgICAgICAgICAgIlB1Y
  mxpYyI6ICJPbGViSVA0Y3otOEVWV05lRFBVTE9rajJlczd2dTdBUF9OTEo1MWs
  xSDBYb1N0UWtFaWtQCiAgQ19vM3FCVmJsVUh2Uk0wblBFQmxjdGdBIn19fX19",
      {
        "signatures": [{
```

```
          "alg": "SHA2",
          "kid": "MB3Z-K5HL-EKAB-6M5J-3PKY-BIYB-JYK5",
          "signature": "Ten3iq7kUJdif_NE94cr-RJ0OJD8qrG-9gj7eu-M6ae
              oqLk3_
  Ew9MqJ8nxvpI13YWEq1Zb9uRfaAbcOghmncfeN9rwTeoApfGEE5lZ5O2h97s1m
  inEKmUEWMOoT2-CBkVSZlH50rYADhbJ9N1ZpWRDAA"}],
        "PayloadDigest": "2K1y20hWCmxcFb_DzTRpHlEyPbqySb-TPF8cA6Y_T5i
            fc
  OupwDkIwlLqWQBML9IxCZyqtZ37vJh4DkAA6LGWWg"}]}}
```


An account registration is deleted using the DeleteAccount transaction.

7. Container Synchronization

All the state associated with a Mesh profile is stored as a sequence of DARE Messages in a Dare Container. The Mesh Service holding the master copy of the persistence stores and the devices connected to the profile containing complete copies (replicas) or partial copies (redactions).

Thus, the only primitive needed to achieve synchronization of the profile state are those required for synchronization of a DARE Container. These steps are:

- Obtain the status of the catalogs and spools associated with the account.
- Download catalog and spool updates
- Upload catalog updates.

To ensure a satisfactory user experience, Mesh Messages are intentionally limited in size to 64 KB or less, thus ensuring that an application can retrieve the most recent 100 messages almost instantaneously on a high bandwidth connection and without undue delay on a slower one.

7.1. Status Transaction

The status transaction returns the status of the containers the device is authorized to access for the specified account together with the updated Device Connection Entry if this has been modified since the entry presented to authenticate the request was issued.

7.2. Download Transaction

The download transaction returns a collection of entries from one or more containers associated with the profile.

Optional filtering criteria **MAY** be specified to only return objects matching specific criteria and/or only return certain parts of the selected messages.

The service **MAY** limit the number of entries returned in an individual response for performance reasons.

7.2.1. Conflict Detection

Clients **SHOULD** check to determine if updates to a container conflict with pending updates on the device waiting to be uploaded. For example, if a contact that the user modified on the device attempting to synchronize was subsequently deleted.

The means of resolving such conflicts is not in the scope of this specification.

7.2.2. Filtering

Clients may request container updates be filtered to redact catalog entries that have been updated or deleted or spool entries that have been read, deleted or were received before a certain date.

7.3. Upload Transaction

The upload transaction upload objects to a catalog or spool.

Multiple objects **MAY** be uploaded at once. Object updates **MAY** be conditional on the successful completion of other upload requests.

The transaction **MAY** be performed in one request/response round trip or with separate round trips to confirm that the transaction is accepted by the service before sending large number of updates.

8. Device Connection

Devices request connection to a Mesh profile using the Connect transaction. Three connection mechanisms are currently defined. All three of which offer strong mutual authentication.

Device Authenticated

Pin Authenticated

EARL Connection Mode

The first two of these mechanisms are initiated from the device being connected which requires that the Mesh Service Account it is being connected to be entered into it. Use of these mechanisms thus requires keyboard and display affordances or accessibility equivalents.

The last mechanism is initiated from an administration device that is already connected to the account. It is intended for use in circumstances where the device being connected does not have the necessary affordances to allow the Device or PIN authenticated modes.

In either case, the connection request is completed by the device requesting synchronization with the Mesh Account using its device credential for authentication. If the connection request was accepted, the device will be provisioned with the Device Connection Assertion allowing it to complete the process.

The Device Connection Assertion includes an overlay device profile containing a set of private key contributions to be used to perform key cogeneration on the original set of device keys to create a new device profile to be used for all purposes associated with the Mesh Profile to which it has just been connected. This assures the user that the keys the device uses for performing operation in the context of their profile are not affected by any compromise that might have occurred during manufacture or at any point after up to the time it was connected to their profile.

8.1. Device Authenticated

The direct connection mechanism requires that both the administration device and the device originating the connection request have data entry and output affordances and that it is possible for the user to compare the authentication codes presented by the two devices to check that they are identical.

8.2. PIN Authenticated

The PIN Connection mechanism is similar to the Direct connection mechanism except that the process is initiated on an administration device by requesting assignment of a new authentication PIN. The PIN is then input to the connecting device to authenticate the request.

8.3. EARL connection mode

The EARL/QR code connection mechanisms are used to connect a constrained device to a Mesh profile by means of an Encrypted Authenticated Resource Locator, typically presented as a QR code on the device itself or its packaging.

9. Mesh Messaging

Mesh Messages provide a means of communication between Mesh Service Accounts with capabilities that are not possible or poorly supported in traditional SMTP mail messaging:

- *End-to-end confidentiality and authentication by default.
- *Abuse mitigation by applying access control to every inbound and outbound message.
- *End-to-end secure group messaging.
- *Transfer of very large data sets (Terabytes).

Note that although Mesh Messaging is designed to facilitate the transfer of very large data sets, the size of Mesh Messages themselves is severely restricted. The current default maximum size being 64 KB. This approach allows Mesh

In addition, the platform anticipates but does not currently support additional cryptographic security capabilities:

- *Traffic analysis resistance using mix networks (Chaum).
- *Simultaneous contract binding using fair contract signing (Micali).

While these capabilities might in time cause Mesh Messaging to replace SMTP, this is not a near term goal. The short-term goal of Mesh Messaging is to support the Contact Exchange and Confirmation applications.

Two important classes of application that are not currently supported directly are payments and presence. While prototypes of these applications have been considered, it is not clear if these are best implemented as special cases of the Confirmation and Contact Exchange applications or as separate applications in their own right.

9.1. Message Exchange

To enable effective abuse mitigation, Mesh Messaging enforces a four corner communication model in which all outbound and inbound messages pass through a Mesh Service which accredits and authorizes the messages on the user's behalf.

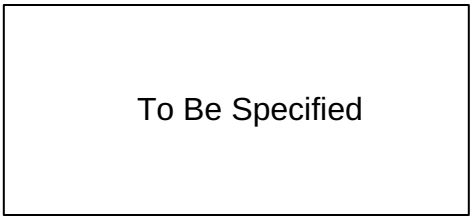

Figure 1

The Post transaction is used for client-service and service-service messaging transactions.

9.1.1. Client-Service (Post Transaction)

To send a message, the client creates the Mesh Message structure, encapsulates it in a DARE Message and forwards this to its service using a Post transaction.

The Post transaction is authenticated to the service by device using the usual means of profile or ticket authentication.

The DARE Message **MUST** be signed under a device signature key accredited by a Device Connection Assertion provided in the message signature block.

9.1.2. Service-Service (Post Transaction)

The Mesh Service receiving the message from the user's device **MAY** attempt immediate retransmission or queue it to be sent at a future time. Mesh Services **SHOULD** forward messages without undue delay.

The Post transaction forwarding the message to the destination service carries the same payload as the original request but is authenticated by the service forwarding it. This authentication **MAY** be my means of either profile or ticket authentication.

9.1.2.1. Denial of Service Mitigation

Services **SHOULD** implement Denial of Service mitigation strategies including limiting the maximum time taken to complete a transaction and refusing connections from clients that engage in patterns of behavior consistent with abuse.

The limitation in message size allows Mesh Services to aggressively time out connections that take too long to complete a transaction. A Mesh Service that hosted on a 10Mb/s link should be able to transfer 20 messages a second. If the service is taking more than 5 seconds to complete a transaction, either the source or the destination service is overloaded or the message itself is an attack.

Imposing hard constraints on Mesh Service performance requires deployments to scale and apply resources appropriately. If a service is attempting to transfer 100 messages simultaneously and 40% are taking 4 seconds or more, this indicates that the number of simultaneous transfers being attempted should be reduced. Contrawise, if 90% are completinin less than a second, the number of threads allocated to sending outbound messages might be increased.

9.1.2.2. Access Control

The inbound service **MUST** subject inbound messages to Access Control according to the credentials presented in the DARE Message payload.

After verifying the signature and checking that the key is properly accredited in accordance with site policy, the service applies authorization controls taking account of:

- *The accreditation of the sender
- *The accreditation of the transmitting Service
- *The type of Mesh Message being sent
- *User policy as specified in their Contact Catalog
- *Site policy.

9.1.3. Service-Client (Synchronization)

The final recipient receives the message by synchronizing their inbound spool.

10. Protocol Schema

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

Every Mesh Portal Service transaction consists of exactly one request followed by exactly one response. Mesh Service transactions MAY cause modification of the data stored in the Mesh Service or the Mesh itself but do not cause changes to the connection state. The protocol itself is thus idempotent. There is no set sequence in which operations are required to be performed. It is not necessary to perform a Hello transaction prior to any other transaction.

10.1. Request Messages

A Mesh Portal Service request consists of a payload object that inherits from the MeshRequest class. When using the HTTP binding, the request MUST specify the portal DNS address in the HTTP Host field.

10.1.1. Message: MeshRequest

Base class for all request messages.

[No fields]

10.1.2. Message: MeshRequestUser

Base class for all request messages made by a user.

Inherits: MeshRequest The fully qualified account name (including
Account: String (Optional) DNS address) to which the request is directed.

DeviceProfile: DareEnvelope (Optional) Device profile of the device making the request.

10.2. Response Messages

A Mesh Portal Service response consists of a payload object that inherits from the MeshResponse class. When using the HTTP binding, the response SHOULD report the Status response code in the HTTP response message. However the response code returned in the payload object MUST always be considered authoritative.

10.2.1. Message: MeshResponse

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

[No fields]

10.3. Imported Objects

The Mesh Service protocol makes use of JSON objects defined in the JOSE Signatgure and Encryption specifications and in the DARE Data At Rest Encryption extensions to JOSE.

10.4. Common Structures

The following common structures are used in the protocol messages:

10.4.1. Structure: KeyValue

Describes a Key/Value structure used to make queries for records matching one or more selection criteria.

Key: String (Optional) The data retrieval key.

Value: String (Optional) The data value to match.

10.4.2. Structure: ConstraintsSelect

Specifies constraints to be applied to a search result. These allow a client to limit the number of records returned, the quantity of data returned, the earliest and latest data returned, etc.

Container: String (Optional) The container to be searched.

IndexMin: Integer (Optional) Only return objects with an index value that is equal to or higher than the value specified.

IndexMax: Integer (Optional) Only return objects with an index value that is equal to or lower than the value specified.

NotBefore: DateTime (Optional) Only data published on or after the specified time instant is requested.

Before: DateTime (Optional) Only data published before the specified time instant is requested. This excludes data published at the specified time instant.

PageKey: String (Optional) Specifies a page key returned in a previous search operation in which the number of responses exceeded the specified bounds.

When a page key is specified, all the other search parameters except for MaxEntries and MaxBytes are ignored and the service returns the next set of data responding to the earlier query.

10.4.3. Structure: ConstraintsData

Specifies constraints on the data to be sent.

MaxEntries: Integer (Optional) Maximum number of entries to send.

BytesOffset: Integer (Optional) Specifies an offset to be applied to the payload data before it is sent. This allows large payloads to be transferred incrementally.

BytesMax: Integer (Optional) Maximum number of payload bytes to send.

Header: Boolean (Optional) Return the entry header

Payload: Boolean (Optional) Return the entry payload

Trailer: Boolean (Optional) Return the entry trailer

10.4.4. Structure: PolicyAccount

Describes the account creation policy including constraints on account names, whether there is an open account creation policy, etc.

Minimum: Integer (Optional) Specifies the minimum length of an account name.

Maximum: Integer (Optional) Specifies the maximum length of an account name.

InvalidCharacters: String (Optional) A list of characters that the service does not accept in account names. The list of characters MAY not be exhaustive but SHOULD include any illegal characters in the proposed account name.

10.4.5. Structure: ContainerStatus

Container: String (Optional)
Index: Integer (Optional)
Digest: Binary (Optional)

10.4.6. Structure: ContainerUpdate

Inherits: ContainerStatus The entries to be uploaded.
Envelopes: DareEnvelope [0..Many]

10.5. Transaction: Hello

Request: HelloRequest
Response: MeshHelloResponse Report service and version information.

The Hello transaction provides a means of determining which protocol versions, message encodings and transport protocols are supported by the service.

The PostConstraints field MAY be used to advise senders of a maximum size of payload that MAY be sent in an initial Post request.

10.5.1. Message: MeshHelloResponse

ConstraintsUpdate: ConstraintsData (Optional) Specifies the default data constraints for updates.

ConstraintsPost: ConstraintsData (Optional) Specifies the default data constraints for message senders.

PolicyAccount: PolicyAccount (Optional) Specifies the account creation policy

EnvelopedProfileService: DareEnvelope (Optional) The enveloped master profile of the service.

EnvelopedProfileHost: DareEnvelope (Optional)
The enveloped profile of the host.

10.6. Transaction: CreateAccount

Request: CreateRequest
Response: CreateResponse Request creation of a new service account or group.

Attempt

10.6.1. Message: CreateRequest

Request binding of an account to a service address.

Inherits: MeshRequest The service account to bind to.
AccountAddress: String (Optional)
SignedProfileMesh: DareEnvelope (Optional) The persistent profile that will be used to validate changes to the account assertion.

SignedAssertionAccount: DareEnvelope (Optional) The signed assertion describing the account.

10.6.2. Message: CreateResponse

Inherits: MeshResponse
Reports the success or failure of a Create transaction.

Reason: String (Optional) Text explaining the status of the creation request.

URL: String (Optional) A URL to which the user is directed to complete the account creation request.

10.7. Transaction: DeleteAccount

Request: DeleteRequest
Response: DeleteResponse Request deletion of a service account.

10.7.1. Message: DeleteRequest

Request creation of a new portal account. The request specifies the requested account identifier and the Mesh profile to be associated with the account.

Inherits: MeshRequestUser
[No fields]

10.7.2. Message: DeleteResponse

Inherits: MeshResponse
Reports the success or failure of a Delete transaction.

[No fields]

10.8. Transaction: Complete

Request: CompleteRequest
Response: CompleteResponse

10.8.1. Message: CompleteRequest

Inherits: StatusRequest
AccountAddress: String (Optional)
ResponseID: String (Optional)

10.8.2. Message: CompleteResponse

Inherits: MeshResponse
SignedResponse: DareEnvelope (Optional)
The signed assertion describing the result of the connect request

10.9. Transaction: Status

Request: StatusRequest
Response: StatusResponse

10.9.1. Message: StatusRequest

Inherits: MeshRequestUser
DeviceUDF: String (Optional)
ProfileMasterDigest: Binary (Optional)
Catalogs: String [0..Many]
Spools: String [0..Many]

10.9.2. Message: StatusResponse

Inherits: MeshResponse
EnvelopedProfileMaster: DareEnvelope (Optional)
The master profile that provides the root of trust for this Mesh

EnvelopedCatalogEntryDevice: DareEnvelope (Optional)
The catalog device entry

ContainerStatus: ContainerStatus [0..Many]

10.10. Transaction: Download

Request: DownloadRequest
Response: DownloadResponse
Request objects from the specified container with the specified search criteria.

10.10.1. Message: DownloadRequest

Inherits: MeshRequestUser
Request objects from the specified container(s).

A client MAY request only objects matching specified search criteria be returned and MAY request that only specific fields or parts of the payload be returned.

Select: ConstraintsSelect [0..Many] Specifies constraints to be applied to a search result. These allow a client to limit the number of records returned, the quantity of data returned, the earliest and latest data returned, etc.

ConstraintsPost: ConstraintsData (Optional) Specifies the data constraints to be applied to the responses.

10.10.2. Message: DownloadResponse

Inherits: MeshResponse
Return the set of objects requested.

Services SHOULD NOT return a response that is disproportionately large relative to the speed of the network connection without a clear indication from the client that it is relevant. A service MAY limit the number of objects returned. A service MAY limit the scope of each response.

Updates: ContainerUpdate [0..Many] The updated data

10.11. Transaction: Upload

Request: UploadRequest
Response: UploadResponse Request objects from the specified container with the specified search criteria.

10.11.1. Message: UploadRequest

Inherits: MeshRequestUser
Upload entries to a container. This request is only valid if it is issued by the owner of the account

Updates: ContainerUpdate [0..Many] The data to be updated

Self: DareEnvelope [0..Many] Entries to be added to the inbound spool on the account, e.g. completion messages.

10.11.2. Message: UploadResponse

Inherits: MeshResponse
Response to an upload request.

Entries: EntryResponse [0..Many]
The responses to the entries.

ConstraintsData: ConstraintsData (Optional) If the upload request contains redacted entries, specifies constraints that apply to the redacted entries as a group. Thus the total payloads of all the messages must not exceed the specified value.

10.11.3. Structure: EntryResponse

IndexRequest: Integer (Optional) The index value of the entry in the request.

IndexContainer: Integer (Optional) The index value assigned to the entry in the container.

Result: String (Optional) Specifies the result of attempting to add the entry to a catalog or spool. Valid values for a message are 'Accept', 'Reject'. Valid values for an entry are 'Accept', 'Reject' and 'Conflict'.

ConstraintsData: ConstraintsData (Optional) If the entry was redacted, specifies constraints that apply to the redacted entries as a group. Thus the total payloads of all the messages must not exceed the specified value.

10.12. Transaction: Publish

Request: PublishRequest
Response: PublishResponse Request to post to a spool from an external party. The request and response messages are extensions of the corresponding messages for the Upload transaction. It is expected that additional fields will be added as the need arises.

10.12.1. Message: PublishRequest

Inherits: MeshRequest The entries to be published. These may
Publications: CatalogedPublication [0..Many] contain the full data or just the identifier, length and fingerprint.

10.12.2. Message: PublishResponse

Inherits: MeshResponse
[No fields]

10.13. Transaction: Post

Request: PostRequest
Response: PostResponse

Request to post to a spool from an external party. The request and response messages are extensions of the corresponding messages for the Upload transaction. It is expected that additional fields will be added as the need arises.

10.13.1. Message: PostRequest

Inherits: MeshRequest The account(s) to which the request is
Accounts: String [0..Many] directed.

Message: DareEnvelope [0..Many] The entries to be uploaded. These MAY be either complete messages or redacted messages. In either case, the messages MUST conform to the ConstraintsUpdate specified by the service

Self: DareEnvelope [0..Many] Messages to be appended to the user's self spool. this is typically used to post notifications to the user to mark messages as having been read or responded to.

10.13.2. Message: PostResponse

Inherits: UploadResponse

[No fields]

10.14. Transaction: Connect

Request: ConnectRequest
Response: ConnectResponse Request information necessary to begin making a connection request.

10.14.1. Message: ConnectRequest

Inherits: MeshRequest The connection request generated by the
MessageConnectionRequestClient: DareEnvelope (Optional) client

10.14.2. Message: ConnectResponse

Inherits: MeshResponse The connection request generated by the
EnvelopedConnectionResponse: DareEnvelope (Optional) client

EnvelopedProfileMaster: DareEnvelope (Optional) The master profile that provides the root of trust for this Mesh

EnvelopedAccountAssertion: DareEnvelope (Optional) The current account assertion

10.15. Transaction: Claim

Request: ClaimRequest
Response: ClaimResponse Claim a publication

10.15.1. Message: ClaimRequest

Inherits: MeshRequest The claim message
EnvelopedMessageClaim: DareEnvelope (Optional)

10.15.2. Message: ClaimResponse

Inherits: MeshResponse The encrypted device profile
CatalogedPublication: CatalogedPublication (Optional)

10.16. Transaction: PollClaim

Request: PollClaimRequest
Response: PollClaimResponse Check party making claim

10.16.1. Message: PollClaimRequest

Inherits: MeshRequest The envelope identifier formed from the
PublicationId: String (Optional) PublicationId.

TargetAccountAddress: String (Optional) Account to which the claim is directed

10.16.2. Message: PollClaimResponse

Inherits: MeshResponse The claim message
EnvelopedMessageClaim: DareEnvelope (Optional)

10.17. Transaction: CreateGroup

Request: CreateGroupRequest
Response: CreateGroupResponse Check party making claim

10.17.1. Message: CreateGroupRequest

Inherits: MeshRequest The service account to bind to.
AccountAddress: String (Optional)
SignedProfileGroup: DareEnvelope (Optional) The persistent profile that will be used to validate changes to the account assertion.

10.17.2. Message: CreateGroupResponse

Inherits: CreateResponse
[No fields]

10.17.3. Structure: CryptographicOperation

KeyId: String (Optional) The key identifier

KeyCoefficient: Binary (Optional) Lagrange coefficient multiplier to be applied to the private key

10.17.4. Structure: CryptographicOperationSign

Inherits: CryptographicOperation The data to sign
Data: Binary (Optional)
PartialR: Binary (Optional) Contribution to the R offset.

10.17.5. Structure: CryptographicOperationKeyAgreement

Inherits: CryptographicOperation
[No fields]

10.17.6. Structure: CryptographicOperationGenerate

Inherits: CryptographicOperation
[No fields]

10.17.7. Structure: CryptographicOperationShare

Inherits: CryptographicOperation
Threshold: Integer (Optional)
Shares: Integer (Optional)

10.17.8. Structure: CryptographicResult

Error: String (Optional)

10.17.9. Structure: CryptographicResultKeyAgreement

Inherits: CryptographicResult
[No fields]

10.18. Transaction: Operate

Request: OperateRequest
Response: OperateResponse Perform a set of cryptographic operations

10.18.1. Message: OperateRequest

Inherits: MeshRequest The service account the capability is bound
AccountAddress: String (Optional) to

10.18.2. Message: OperateResponse

Inherits: MeshResponse
[No fields]

11. Security Considerations

The security considerations for use and implementation of Mesh services and applications are described in the Mesh Security Considerations guide [draft-hallambaker-mesh-security].

12. IANA Considerations

All the IANA considerations for the Mesh documents are specified in this document

13. Acknowledgements

A list of people who have contributed to the design of the Mesh is presented in [draft-hallambaker-mesh-architecture].

14. Normative References

[draft-hallambaker-mesh-architecture]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part I: Architecture Guide", Work in Progress, Internet-Draft, draft-hallambaker-mesh-architecture-13, 9 March 2020, <https://tools.ietf.org/html/draft-hallambaker-mesh-architecture-13>.

[draft-hallambaker-mesh-security]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part VII: Security Considerations", Work in Progress, Internet-Draft, draft-hallambaker-mesh-security-04, 9 March 2020, <https://tools.ietf.org/html/draft-hallambaker-mesh-security-04>.

[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>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/rfc/rfc3339>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/rfc/rfc7230>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

15. Informative References

[draft-hallambaker-mesh-developer]
Hallam-Baker, P., "Mathematical Mesh: Reference Implementation", Work in Progress, Internet-Draft, draft-hallambaker-mesh-developer-09, 23 October 2019, <https://tools.ietf.org/html/draft-hallambaker-mesh-developer-09>.

[ECMA-262] Ecma International, "ECMAScript(R) 2017 Language Specification", June 2017.