Network Working Group
Internet-Draft
Intended status: Informational
Expires: March 4, 2019

P. Hallam-Baker
Comodo Group Inc.

August 31, 2018

Mathematical Mesh Part II: Reference
draft-hallambaker-mesh-reference-10

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.

This document is also available online at http://mathmesh.com/Documents/draft-hallambaker-mesh-reference.html [1] .

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 March 4, 2019.

Table of Contents

1. Introduction 4
2. Definitions 5
2.1. Requirements Language 5
2.2. Defined Terms 5
2.3. Related Specifications 5
2.4. Implementation Status 5
3. Mesh Profiles 6
3.1. Mesh Master Profile 6
3.2. Mesh Device Profile 7
3.3. Mesh Personal Profile 7
3.4. Mesh Application Profile 8
4. Mesh Portal Service 8
4.1. Creating a Mesh Service Account 8
4.2. Using Recovery Records 9
4.2.1. Creating a Recovery Record 9
4.2.2. Recovering a Profile. 11
4.3. Connecting a Device to a Portal Account 11
4.3.1. Deleting a Portal Account 13
5. Mesh Catalogs 14
5.1. Synchronizing a Device to a Catalog 14
6. Mesh Catalog Services 16
6.1. The Contacts Catalog 16
6.2. Credential Catalog 16
6.3. Tasks Catalog 16
6.4. Mail Catalog 17
6.5. SSH Catalog 17
6.6. Recryption Catalog 17
7. Mesh Messaging 17
7.1. Message Origination 18
7.2. Message Transit 19
7.3. Receiving Messages 20
7.3.1. Responding to Messages 21
8. Messaging Services 21
8.1. Contact Messaging Service 21
8.2. Confirmation Messaging Service 21
8.3. Asynchronous Messaging Service 21
8.4. Synchronous Messaging Service 21
9. Shared Classes 21
9.1. Cryptographic Data Classes 21
9.1.1. Structure: PublicKey 22
9.1.2. Structure: SignedData 22

9.1.3. Structure: EncryptedData 22
9.2. Common Application Classes 22
9.2.1. Structure: Connection 22
10. Mesh Profile Objects 23
10.1. Base Profile Objects 23
10.1.1. Structure: Entry 23
10.1.2. Structure: SignedProfile 23
10.1.3. Structure: Advice 23
10.1.4. Structure: PortalAdvice 24
10.1.5. Structure: Profile 24
10.2. Device Profile Classes 24
10.2.1. Structure: SignedDeviceProfile 24
10.2.2. Structure: DeviceProfile 24
10.2.3. Structure: DevicePrivateProfile 25
10.3. Master Profile Objects 25
10.3.1. Structure: SignedMasterProfile 25
10.3.2. Structure: MasterProfile 25
10.4. Personal Profile Objects 26
10.4.1. Structure: SignedPersonalProfile 26
10.4.2. Structure: PersonalProfile 26
10.4.3. Structure: ApplicationProfileEntry 26
10.5. Application Profile Objects 27
10.5.1. Structure: SignedApplicationProfile 27
10.5.2. Structure: ApplicationProfile 27
10.5.3. Structure: ApplicationProfilePrivate 27
10.5.4. Structure: ApplicationDevicePublic 27
10.5.5. Structure: ApplicationDevicePrivate 28
10.6. Key Escrow Objects 28
10.6.1. Structure: EscrowEntry 28
10.6.2. Structure: OfflineEscrowEntry 28
10.6.3. Structure: OnlineEscrowEntry 28
10.6.4. Structure: EscrowedKeySet 28
11. Portal Connection 28
11.1. Connection Request and Response Structures 28
11.1.1. Structure: ConnectionRequest 29
11.1.2. Structure: SignedConnectionRequest 29
11.1.3. Structure: ConnectionResult 29
11.1.4. Structure: SignedConnectionResult 29
12. Mesh Portal Service Reference 29
12.1. Request Messages 30
12.1.1. Message: MeshRequest 30
12.2. Response Messages 30
12.2.1. Message: MeshResponse 30
12.3. Imported Objects 30
12.4. Common Structures 30
12.4.1. Structure: KeyValue 30
12.4.2. Structure: SearchConstraints 31
12.5. Transaction: Hello 31

12.6. Transaction: ValidateAccount 31
12.6.1. Message: ValidateRequest 32
12.6.2. Message: ValidateResponse 32
12.7. Transaction: CreateAccount 33
12.7.1. Message: CreateRequest 33
12.7.2. Message: CreateResponse 33
12.8. Transaction: DeleteAccount 33
12.8.1. Message: DeleteRequest 34
12.8.2. Message: DeleteResponse 34
12.9. Transaction: Get 34
12.9.1. Message: GetRequest 34
12.9.2. Message: GetResponse 35
12.10. Transaction: Publish 35
12.10.1. Message: PublishRequest 35
12.10.2. Message: PublishResponse 36
12.11. Transaction: Status 36
12.11.1. Message: StatusRequest 36
12.11.2. Message: StatusResponse 36
12.12. Transaction: ConnectStart 37
12.12.1. Message: ConnectStartRequest 37
12.12.2. Message: ConnectStartResponse 37
12.13. Transaction: ConnectStatus 37
12.13.1. Message: ConnectStatusRequest 38
12.13.2. Message: ConnectStatusResponse 38
12.14. Transaction: ConnectPending 38
12.14.1. Message: ConnectPendingRequest 38
12.14.2. Message: ConnectPendingResponse 39
12.15. Transaction: ConnectComplete 39
12.15.1. Message: ConnectCompleteRequest 39
12.15.2. Message: ConnectCompleteResponse 39
12.16. Transaction: Transfer 40
12.16.1. Message: TransferRequest 40
12.16.2. Message: TransferResponse 40
13. Security Considerations 40
14. IANA Considerations 41
15. Acknowledgements 41
16. References 41
16.1. Normative References 41
16.2. Informative References 41
16.3. URIs 41
Author's Address 41

1. Introduction

This document describes the data structures and network protocols of the Mathematical Mesh illustrated with illustrative examples. For an overview of the Mesh objectives and architecture, consult the accompanying Architecture Guide [draft-hallambaker-mesh-architecture]

This document has two main sections. The first section presents examples of using the Mesh to address common use cases. The second section contains the Mesh profile and service schemas. All the material in both sections is generated from the Mesh reference implementation [draft-hallambaker-mesh-developer] .

Although some of the services described in this document could be used to replace existing Internet protocols including FTP and SMTP, 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 Profiles

Every Mesh user has a Mesh profile which contains all the configuration information for all their devices and all their network services. For convenience, the mesh profile is divided into four separate parts, the Master profile, the Personal Profile, Device Profiles and Application Profiles as follows:

3.1. Mesh Master Profile

The Mesh Master Profile describes the criteria for validating an owner's personal profile. In particular, the master profile specifies the Master Signature Key that is used as the root of trust under which the master profile is validated and a set of Administration Signature Keys under which the personal profile is validated.

Master Signature Key is immutable. By definition, it is not possible to change the Master Signature Key without creating a new master profile.

The UDF fingerprint of the Master Signature Key is the fingerprint of the Master Profile and the Personal Profile created underneath it.

For example, Alice creates the following Master Profile, it has a Master Signature Key and a Master Recovery Key. There is one administration device specified, the correcponding device profile is described in the next section.

{AliceMasterProfile}

Figure 1

The UDF fingerprint of Alice's Master signature key is:

{AliceFingerprint}

Figure 2

A Master Profile MAY be revoked but never expires. It is the intended that a user should not normally need to change their master profile.

The only means of expiring a master profile that is currently supported is to sign a 'suicide note' for the profile. This is an assertion that the master profile is invalid that has been signed by the user. An application MAY generate such a suicide note at the

time that the master profile is created and archive it so that the profile owner's executors can revoke the profile after death.

{AliceMasterProfileSuicide}

Figure 3

Since a Master Signature Key is immutable, no provision is made for modifying a Master Signature Key, nor is such provision possible. Should a user lose control of the private keys listed in their master profile, the only remediation possible is to create a new Master Signature Key and master profile and then persuade parties relying on the original that it is the successor.

3.2. Mesh Device Profile

To make use of the Mesh Profile, Alice needs to connect at least one device. Every device profile has an encryption, signature and authentication key.

Alice decides to use her desktop personal computer as her first administration device. Her device profile is:

{AliceDeviceProfile}

Figure 4

Note that each of the keys is a Diffie-Hellman Key. This enables the use of distributed key generation techniques as described in part III. These will be transitioned to Elliptic Curve Diffie Hellman keys for production use.

3.3. Mesh Personal Profile

Alice's personal profile contains her master profile and a list of device profiles. It is signed by her administration device using its signature key.

Alice's personal profile specifies her master profile and the device profile of her personal computer:

{AlicePersonalProfile}

Figure 5

A personal profile instance MUST specify the device profile of the administration profile that signed that instance.

3.4. Mesh Application Profile

Alice also creates one or more application profiles, each of which are signed by her administration key.

Alice creates a credential catalog to allow her to create strong passwords with a work factor of 2^128 and use them on multiple devices, in this case, her administration device and her

{AliceApplicationProfile}

Figure 6

4. Mesh Portal Service

The Mesh Portal Service is the subset of Mesh Service operations that manage Mesh profiles. A Mesh Service MUST support the Mesh Portal Service but is not required to support any other service.

4.1. Creating a Mesh Service Account

Having created a personal profile, Alice requests creation of an account at a Mesh Service. The first step in this process is choosing a Mesh Service account address 'Mesh address'

A Mesh address has the format user@domain where domain is the DNS name of the Mesh service and user is the name of their account at that service.

Services MAY support the use of any unicode character sequence permitted for use as an SMTP email address by RFC6530. Matching of Mesh addresses is case insensitive for latin characters (a-z, A-Z) but no similar mappings are supported for other character sets.

Alice selects the Mesh Service 'example.com' and the name 'alice'. Her Mesh client first checks to see if the name is available:

Request {Verify alice@example.com}

Figure 7

The Mesh service responds stating that the address is available.

The ValidateRequest message contains the requested account identifier and an optional language parameter to allow the service to provide informative error messages in a language the user understands. The Language field contains a list of ISO language identifier codes in order of preference, most preferred first.

Response {Verify alice@example.com}

Figure 8

The ValidateResponse message returns the result of the validation request in the Valid field. Note that even if the value true is returned, a subsequent account creation request MAY still fail.

[Note that for the sake of concise presentation, the HTTP binding information is omitted from future examples.]

The Mesh client requests that the account be created and bound to the (provided) personal profile:

Request {Account Create alice@example.com}

Figure 9

The Mesh service responds stating that the address is available:

Response {Account Create alice@example.com}

Figure 10

4.2. Using Recovery Records

Before using her newly created profile, Alice makes sure that she can recover it in the case of a catastrophe. She also wants to make sure that her master profile won't be compromised if the machine she created it on is compromised by deleting the key information from the machine. To do this, she creates a Recovery Record.

A recovery record contains the private keys associated with her master profile encrypted using a strong symmetric cipher (AES 256 in this case). Recovery records are indexed by means of the UDF fingerprint derrived from the decryption key. Thus, knowledge of the decryption key is sufficient to locate the recovery record in a collection of recovery records and knowledge of the index is evidence that a requestor knows the decryption key.

4.2.1. Creating a Recovery Record

The plaintext of the recovery record specifies the private keys associated with the Master Signature Key and Master Escrow Key:

{Recovery RecordPlaintext}

Figure 11

A Master Recovery Key is created. In this case, Alice is using a Master Recovery Key of 128 bits so that the recovery key shares are as compact as possible.

{MasterRecoveryKey}

Figure 12

The HKDF function is used to derrive the Encryption Key for the Recovery Record and the Recovery index:

{RecoveryEncryptionKey}
{RecoveryIndex}

Figure 13

The Recovery record is encrypted using the DARE Message Format

{Recovery RecordDARE}

Figure 14

The Mesh client then creates an authenticated request to post the recovery record to the profile:

{AuthenticatedRecoveryRequest}

Figure 15

The Mesh Service returns its response:

{AuthenticatedRecoveryResponse}

Figure 16

[Note that for the sake of concise presentation, the request and response authentication information is omitted from future examples.]

Having successfully posted the recovery data to the service, the client presents Alice with a list of recovery shares that can be used to recover the data. The calculation of the recovery shares is described in part III.

{Recovery shares 2 of 3}

Figure 17

4.2.2. Recovering a Profile.

To test her ability to recover her master profile, Alice deletes her master profile from her administration device=. To recover her profile, Alice reconstructs the recovery secret from two of her shares and uses this information to request recovery:

{RecoveryRecordRequest}

Figure 18

Note that this request is not authenticated.

The Mesh Service locates the requested data and responds:

{RecoveryRecordResponse}

Figure 19

The client recovers the Master profile information and verifies it, then uses the data to reactivate the

4.3. Connecting a Device to a Portal Account

Connecting a device to a profile requires the client on the new device to interact with a client on a device that has administration capabilities, i.e. it has access to an Online Signing Key. Since clients cannot interact directly with other clients, a service is required to mediate the connection. This service is provided by a Mesh portal provider.

All service transactions are initiated by the clients. First the connecting device posts ConnectStart, after which it may poll for the outcome of the connection request using ConnectStatus.

Periodically, the Administration Device polls for a list of pending connection requests using ConnectPending. After posting a request, the administration device posts the result using ConnectComplete:

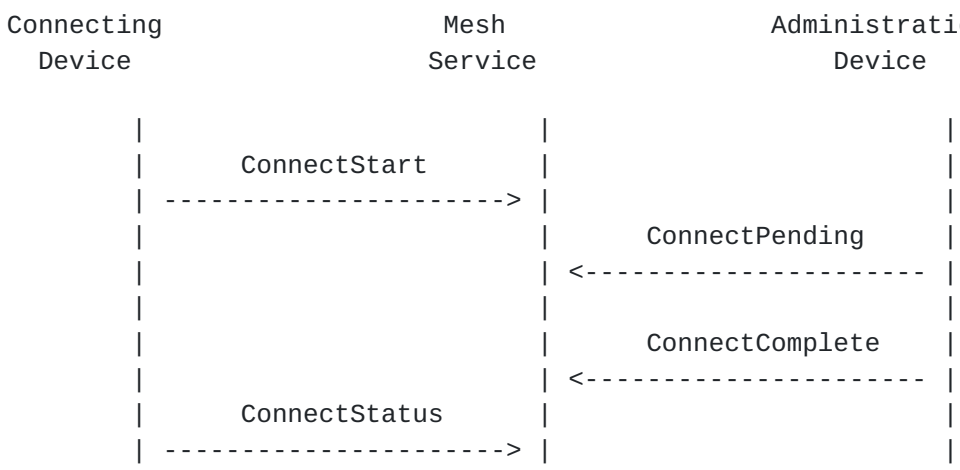

Figure 20

The Device connection flow is actually an example of the Messaging flow in that it is initiated by an untrusted device making a connection request to the Mesh Service which the user's administration device collects and responds to in the same fashion as any other messaging flow.

The process is initiated by a request from the device to post a connection request.

Request {ConnectStart alice@example.com}

Figure 21

The request is accepted. Note that if abuse is a concern, we may have required the use of a one time passcode to validate the request. The service responds with the personal profile bound to the account.

Response {ConnectStart alice@example.com}

Figure 22

The fingerprint of the device profile and the fingerprint of the personal profile are combined to create a request verification code. This is displayed on Alice's device

{Verification code.}

Figure 23

To authorize the request, the administration device begins by synchronizing the connect message spool:

Request {SyncPending Connect alice@example.com}

Figure 24

The service responds with a list of pending requests optionally filtered according to criteria provided by Alice:

Response {SyncPending Connect alice@example.com}

Figure 25

Alice Accepts the request. Her administration device creates and signs a Device Authorization and posts it to the Mesh Service where it is added to the Device Catalog:

Request {ConnectPost alice@example.com}

Figure 26

The request is successful:

Response {ConnectPost alice@example.com}

Figure 27

Finally, the device polls the service and recieves notice that the request has been accepted:

Request {ConnectStatus alice@example.com}

Figure 28

The acceptance includes a copy of the Device Authorization(s).

Response {ConnectStatus alice@example.com}

Figure 29

4.3.1. Deleting a Portal Account

Should she ever decide to stop using the Mesh Service, Alice can request that her account be deleted. Note that this only affects her account on the service and not on her local machine.

{DeleteRequest}

Figure 30

The Mesh Service returns its response:

{DeleteResponse}

Figure 31

Once a Mesh address has been deleted, reuse of the address by a new profile is entirely a matter for site policy. A Mesh Service MAY refuse to allow any request to create an account with a previously used address under any circumstances or MAY allow any party to reuse the address.

Mesh addresses are inherently transient. If a permanent and immutable address is required for some purpose, the Strong Internet Name of the Mesh Address SHOULD be used instead. This name binds the Mesh profile fingerprint to the Mesh Address, thus creating a name that can be regarded as unambiguously identifying the profile owner and means of contact.

5. Mesh Catalogs

A Mesh Catalog Service contains a set of entries that are created by the user for their own use.

A catalog entry MUST be signed by the signature key of a device that is specified in the user's Personal Profile.

Each entry in a Mesh catalog has a unique identifier that acts as its primary key.

Mesh catalogs are typically compact and updated infrequently. Given current storage costs and typical network bandwidth, it is to be expected that most users will be best served by a model in which every device contains a complete copy of the user's catalog(s) that are of interest to it rather than support a model in which connected devices hunt an peck for the desired records on the server. Such an approach is in any case likely to be impossible for the majority of Mesh applications in which the server content is end-to-end encrypted.

5.1. Synchronizing a Device to a Catalog

Synchronization of the catalog data stored on a device with that stored by the Mesh service is bidirectional. Catalog updates stored on the device are merged with those stored on the service and any conflicts reported to the user.

Each device that has the access privilege to update catalog entries thus has two separate queues, one containing a (possibly incomplete) copy of the append-only log held by the service, the other containing updates that have been made on the device but have not yet been accepted by the service.

When a device synchronizes, it:

o Downloads relevant updates from the service to the device.

Devices MAY perform these operations in either order or simultaneously (if the service permits). But regardless of the order in which these are performed by a particular device, there is only one catalog and it is maintaind by the service. Thus all updates that are accepted SHALL be applied to the catalog after all the previous updates.

Since every object has a distinct and independent lifecycle in the Mesh persistence model, detecting a conflict early on in the synchronization process does not invalidate updates to other objects which are independent.

For example, consider the scenario in which Alice synchronizes two devices to her credential catalog.

Alice is already using the password management features of her browser but this service does not provide end-to-end encryption. Alice's Mesh client provides a feature that allows her to export the usernames and passwords from her browser and store them in a Mesh catalog.

Alice's first device creates two credential entries.

{AliceCredential1}

Figure 32

When multiple catalog entries are being encrypted at the same time, these MAY be encrypted under a single key agreement or under a separate key agreement for each entry. Here, a single key agreement is shared:

{AliceCredential1Request}

Figure 33

Since the catalog is empty, the service accepts the update entries and responds with the catalog index before and after the items were accepted.

{AliceCredential1Response}

Figure 34

Alice then attempts to syncrhonize a second device. The browser on the second device has two entries, one of which maches an entry in the first and the other of which is different:

{AliceCredential2}

Figure 35

When the service responds to the second request, the first entry is rejected as a possible conflict and the second is accepted. Note that even though the username/password values are identical, the service does not know this because they are end-to-end encrypted and the service does not have a decryption key. The service responds with a list of the frame numbers of the rejected entries.

{AliceCredential1Response}

Figure 36

Entries are deleted from a catalog with the delete method. The request specifies the catalog to be updated and the list of entries to be deleted:

{AliceDeleteCredential}

Figure 37

6. Mesh Catalog Services

6.1. The Contacts Catalog

{includes recryption membership notifications}

6.2. Credential Catalog

6.3. Tasks Catalog

6.4. Mail Catalog

6.5. SSH Catalog

6.6. Recryption Catalog

The recryption catalog is unique in that it is the only Mesh Service that contains entries that are to be decrypted by the Mesh Service itself

Recryption Group Administrator entries Contain the information that an administrator requires to administer a recryption group. These are encrypted such that only the administrators can decrypt them.

Recryption Group Member entries Contain the information that the Recryption Service requires to respond to recryption requests encrypted under the server key.

7. Mesh Messaging

Mesh messaging services are very similar to Mesh catalog services but with one important difference: Requests to append or update message entries come from a third party that may prove untrustworthy. It is therefore necessary to apply access control to inbound message requests.

The persistence store for a messaging service is called a spool. As with the catalog store of a catalog service, the DARE Container format is designed to support the requirements of managing a messaging service spool but Mesh Services MAY use any persistence technology that meets their needs.

Some Mesh messaging services are standalone while others are closely related to a catalog service.

- Accepting a recryption membership notification SHOULD result in an entry being added to the recipient's credential catalog.

- Accepting a device connection request MUST result in an entry being added to the user's devices catalog.

The PostUpdate method allows a Mesh client to reply to an inbound request and post an entry to the accompanying catalog at the same time.

Mesh messaging services adopt a four corner communication mode in which the sender of a message forwards the request to their own Mesh Service which in turn contacts the recipient's Mesh service to

organize delivery. As with any other four Mesh Service MAY act for both the sender and receiver

The only circumstance in which the sender and recipient communicate directly is in a two phase synchronous protocol in which a four corner first phase is used to negotiate parameters for a direct connection in the second phase.

7.1. Message Origination

Messages are posted to the senders outbound Mesh Service using the Post method.

Alice meets Bob and they 'bump' phones performing a QR code exchange. To begin this exchange, Alice's device generates a random one-time use passcode. Note that since this passcode is only used to authenticate the exchange to mitigate abuse, a work factor of 2^64 is more than sufficient.

```
lYFAVLNJLkC
```

Figure 38

The QR code is:

[QR code image]

The decoded URI is:

```
mmm:alice@example.com.mmm-wekjhwkjehrkjwher:c:lYFAVLNJLkC
```

Figure 39

Bob's phone reads the QR code and creates a contact request message containing Bob's information. The contact request asks to be able to send various types of message.

```
{BobContactPostRequest}
```

Figure 40

Messages are subject to access control by both the inbound and outbound services.

Bob's Mesh Service checks to see if the rate of contact requests he has made in the past is excessive. Finding that it is not, the contact request is accepted and placed in the outbound messages queue.

Bob's service responds with a unique identifier that MAY be used to check on the status of the message:

{BobContactPostRequest}

Figure 41

A short while later, Bob's phone asks for a status update on the request.

{BobContactStatusRequest}

Figure 42

Alice has not responded yet (she is talking to Bob in person).

{BobContactStatusRequest}

Figure 43

If Bob decides not to connect after all, he can cancel the request.

7.2. Message Transit

Passing of messages between Mesh Services is called transit.

To begin a message transfer, the outbound service makes a PostRequest to the inbound service which contains the message headers and the maximum size of the payload.

{OutboundPostRequest}

Figure 44

The inbound service performs access control on the PostRequest according to site policy which MAY include use any resources the inbound service chooses to use, including:

- Valid one time use authorization code issued by the recipient to the sender
- Credentials provided by the inbound service.
- The sender's entry in the recipient's contact catalog.
- The type of access requested.

The access control policy is set by the Mesh Service and the user. When setting an access control policy, both should consider the likelihood that the recipient would wish to accept the message and the risk of harm resulting.

Different users will naturally require different access policies. A celebrity receiving hundreds of contacts a day is likely to require closer access control criteria than a person receiving one request a week. A request to send email messages presents a lower degree of risk than a request to send invoices or program code.

In this case, the request has been pre-approved by means of a one time use authentication code provided by Alice's device. The inbound service has no means of verifying the authentication code at this stage but accepts the request knowing that it will be rejected by Alice's client if it proves to be bogus.

{OutboundPostResponse}

Figure 45

Since the contact request message is short, the inbound service authorizes the outbound service to send the message body immediately. If the message was longer, the inbound service might tell the outbound to defer delivery of the message body which might be completed by means of an incremental transfer (e.g. in chunks of 4MB at a time).

This mechanism allows the same messaging protocol to be used to transfer messages of a few bytes to multiple terabytes. A Mesh Service is not required to support such messages however and particularly in the case of very large messages, may delgate collection of the message payload to the destination device.

7.3. Receiving Messages

Pending messages are received by synchronizing the message spool of the device with the message spool of the messaging service. This process is identical to synchonizing a catalog.

{SyncRequest}

Figure 46

There is only one message in the contact request spool to be synchronized:

{SyncResponse}

Figure 47

A device MAY improve the user experience by requesting the service provide just the headers of the queued messages or to filter messages of a particular type or which have particular characteristics.

7.3.1. Responding to Messages

As previously noted, the response to a message request frequently entails an update to the user's corresponding catalog. Otherwise, posting a response is the same as a request.

Alice's phone verifies the authentication code on Bob's request and automatically approves it for the level of access Alice specified when they bumped phones. A new contact entry is created together with a contact request message to be returned to Bob notifying him that his request was approved by Alice and providing him with her details for his contact catalog.

{ContactAddResponse}

Figure 48

8. Messaging Services

8.1. Contact Messaging Service

8.2. Confirmation Messaging Service

8.3. Asynchronous Messaging Service

8.4. Synchronous Messaging Service

9. Shared Classes

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

9.1. Cryptographic Data Classes

Most Mesh objects are signed and/or encrypted. For consistency all Mesh classes make use of the cryptographic data classes described in this section.

9.1.1. Structure: PublicKey

The PublicKey class is used to describe public key pairs and trust assertions associated with a public key.

UDF: String (Optional) UDF fingerprint of the public key parameters/

X509Certificate: Binary (Optional) List of X.509 Certificates

X509Chain: Binary [0..Many] X.509 Certificate chain.

X509CSR: Binary (Optional) X.509 Certificate Signing Request.

9.1.2. Structure: SignedData

Container for JOSE signed data and related attributes.

Data: Binary (Optional) The signed data

9.1.3. Structure: EncryptedData

Container for JOSE encrypted data and related attributes.

Data: Binary (Optional) The encrypted data

9.2. Common Application Classes

9.2.1. Structure: Connection

Describes network connection parameters for an application

ServiceName: String (Optional) DNS address of the server

Port: Integer (Optional) TCP/UDP Port number

Prefix: String (Optional) DNS service prefix as described in [!RFC6335]

Security: String [0..Many] Describes the security mode to use. Valid choices are Direct/Upgrade/None

UserName: String (Optional) Username to present to the service for authentication

Password: String (Optional) Password to present to the service for authentication

URI: String (Optional) Service connection parameters in URI format

Authentication: String (Optional) List of the supported/acceptable authentication mechanisms, preferred mechanism first.

TimeOut: Integer (Optional) Service timeout in seconds.

Polling: Boolean (Optional) If set, the client should poll the specified service intermittently for updates.

10. Mesh Profile Objects

10.1. Base Profile Objects

10.1.1. Structure: Entry

Base class for all Mesh Profile objects.

Identifier: String (Optional) Globally unique identifier that remains constant for the lifetime of the entry.

10.1.2. Structure: SignedProfile

Inherits: Entry

Contains a signed profile entry

SignedData: JoseWebSignature (Optional) The signed profile.

Note that each child of SignedProfile requires that the Payload field of the SignedData object contain an object of a specific type. For example, a SignedDeviceProfile object MUST contain a Payload field that contains a DeviceProfile object.

Unauthenticated: Advice (Optional) Additional data that is not authenticated.

10.1.3. Structure: Advice

Additional data bound to a signed profile that is not authenticated.

Default: DateTime (Optional) If specified, the profile was the default profile at the specified date and time. The current default for that type of profile is the profile with the most recent Default timestamp.

10.1.4. Structure: PortalAdvice

Describes the portal(s) at which the profile is registered.

Inherits: Advice

Inherits: Advice

PortalAddress: String [0..Many] A portal address at which this profile is registered.

10.1.5. Structure: Profile

Inherits: Entry

Parent class from which all profile types are derived

Names: String [0..Many] Fingerprints of index terms for profile retrieval. The use of the fingerprint of the name rather than the name itself is a precaution against enumeration attacks and other forms of abuse.

Updated: DateTime (Optional) The time instant the profile was last modified.

NotaryToken: String (Optional) A Uniform Notary Token providing evidence that a signature was performed after the notary token was created.

10.2. Device Profile Classes

10.2.1. Structure: SignedDeviceProfile

Inherits: SignedProfile

Contains a signed device profile

[No fields]

10.2.2. Structure: DeviceProfile

Inherits: Profile

Describes a mesh device.

Description: String (Optional) Description of the device

DeviceSignatureKey: PublicKey (Optional) Key used to sign certificates for the DAK and DEK. The fingerprint of the DSK is the UniqueID of the Device Profile

DeviceAuthenticationKey: PublicKey (Optional) Key used to authenticate requests made by the device.

DeviceEncryptiontionKey: PublicKey (Optional) Key used to pass encrypted data to the device such as a DeviceUseEntry

10.2.3. Structure: DevicePrivateProfile

Private portion of device encryption profile.

DeviceSignatureKey: Key (Optional) Private portion of the DeviceSignatureKey

DeviceAuthenticationKey: Key (Optional) Private portion of the DeviceAuthenticationKey

DeviceEncryptiontionKey: Key (Optional) Private portion of the DeviceEncryptiontionKey

10.3. Master Profile Objects

10.3.1. Structure: SignedMasterProfile

Inherits: SignedProfile

Contains a signed Personal master profile

[No fields]

10.3.2. Structure: MasterProfile

Inherits: Profile

Describes the long term parameters associated with a personal profile.

MasterSignatureKey: PublicKey (Optional) The root of trust for the Personal PKI, the public key of the PMSK is presented as a self-signed X.509v3 certificate with Certificate Signing use enabled. The PMSK is used to sign certificates for the PMEK, POSK and PKEK keys.

MasterEscrowKeys: PublicKey [0..Many] A Personal Profile MAY contain one or more PMEK keys to enable escrow of private keys used for stored data.

OnlineSignatureKeys: PublicKey [0..Many] A Personal profile contains at least one POSK which is used to sign device administration application profiles.

10.4. Personal Profile Objects

10.4.1. Structure: SignedPersonalProfile

Inherits: SignedProfile

Contains a signed Personal current profile

[No fields]

10.4.2. Structure: PersonalProfile

Inherits: Profile

Describes the current applications and devices connected to a personal master profile.

SignedMasterProfile: SignedMasterProfile (Optional) The corresponding master profile. The profile MUST be signed by the PMSK.

Devices: SignedDeviceProfile [0..Many] The set of device profiles connected to the profile. The profile MUST be signed by the DSK in the profile.

Applications: ApplicationProfileEntry [0..Many] Application profiles connected to this profile.

10.4.3. Structure: ApplicationProfileEntry

Personal profile entry describing the privileges of specific devices.

Identifier: String (Optional) The unique identifier of the application

Type: String (Optional) The application type

Friendly: String (Optional) Optional friendly name identifying the application.

AdminDeviceUDF: String [0..Many] List of devices authorized to sign application profiles

DecryptDeviceUDF: String [0..Many] List of devices authorized to read private parts of application profiles.

AccountID: String (Optional) The account at which the profile is located.

10.5. Application Profile Objects

10.5.1. Structure: SignedApplicationProfile

Inherits: SignedProfile

Contains a signed device profile

[No fields]

10.5.2. Structure: ApplicationProfile

Inherits: Profile

Parent class from which all application profiles inherit.

[No fields]

10.5.3. Structure: ApplicationProfilePrivate

Inherits: Entry

The base class for all private profiles.

[No fields]

10.5.4. Structure: ApplicationDevicePublic

Inherits: Entry

Describes the public per device data

DeviceDescription: String (Optional) Description of the device for convenience of the user.

DeviceUDF: String (Optional) Fingerprint of device that this key corresponds to.

10.5.5. Structure: ApplicationDevicePrivate

Inherits: Entry

Describes the private per device data

[No fields]

10.6. Key Escrow Objects

10.6.1. Structure: EscrowEntry

Inherits: Entry

Contains escrowed data

EncryptedData: JoseWebEncryption (Optional) The encrypted escrow data

10.6.2. Structure: OfflineEscrowEntry

Inherits: EscrowEntry

Contains data escrowed using the offline escrow mechanism.

[No fields]

10.6.3. Structure: OnlineEscrowEntry

Inherits: EscrowEntry

Contains data escrowed using the online escrow mechanism.

[No fields]

10.6.4. Structure: EscrowedKeySet

A set of escrowed keys.

[No fields]

11. Portal Connection

11.1. Connection Request and Response Structures

11.1.1. Structure: ConnectionRequest

Describes a connection request.

ParentUDF: String (Optional) UDF of Mesh Profile to which connection is requested.

Device: SignedDeviceProfile (Optional) The Device profile to be connected

11.1.2. Structure: SignedConnectionRequest

Inherits: SignedProfile

Contains a ConnectionRequest signed by the corresponding device signature key.

[No fields]

11.1.3. Structure: ConnectionResult

Describes the result of a connection request.

Inherits: ConnectionRequest

Inherits: ConnectionRequest

Result: String (Optional) The result of the connection request. Valid responses are: Accepted, Refused, Query.

11.1.4. Structure: SignedConnectionResult

Inherits: SignedProfile

Contains a signed connection result

[No fields]

12. Mesh Portal Service Reference

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 Portal 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 a ValidateAccount, Publish or any other transaction.

12.1. 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.

12.1.1. Message: MeshRequest

Base class for all request messages.

Portal: String (Optional) Name of the Mesh Portal Service to which the request is directed.

12.2. 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.

12.2.1. Message: MeshResponse

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

[No fields]

12.3. Imported Objects

The Mesh Service protocol makes use of JSON objects defined in the JOSE Signatgure and Encryption specifications.

12.4. Common Structures

The following common structures are used in the protocol messages:

12.4.1. 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.

12.4.2. Structure: SearchConstraints

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.

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.

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

MaxBytes: Integer (Optional) Maximum number of data bytes to return.

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.

12.5. Transaction: Hello

Request: HelloRequest

Request: HelloRequest

Response: HelloResponse

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.

12.6. Transaction: ValidateAccount

Request: ValidateRequest

Request: ValidateRequest

Response: ValidateResponse

Request validation of a proposed name for a new account.

For validation of a user's account name during profile creation.

12.6.1. Message: ValidateRequest

Inherits: MeshRequest

Describes the proposed account properties. Currently, these are limited to the account name but could be extended in future versions of the protocol.

Account: String (Optional) Account name requested

Reserve: Boolean (Optional) If true, request a reservation for the specified account name. Note that the service is not obliged to honor reservation requests.

Language: String [0..Many] List of ISO language codes in order of preference. For creating explanatory text.

12.6.2. Message: ValidateResponse

Inherits: MeshResponse

States whether the proposed account properties are acceptable and (optional) returns an indication of what properties are valid.

Note that receiving a 'Valid' responseto a Validate Request does not guarantee creation of the account. In addition to the possibility that the account namecould be requested by another user between the Validate and Create transactions, a portal service MAY perform more stringent validation criteria when an account is actually being created. For example, checking with the authoritative list of current accounts rather than a cached copy.

Valid: Boolean (Optional) If true, the specified account identifier is acceptable. If false, the account identifier is rejected.

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.

Reason: String (Optional) Text explaining the reason an account name was rejected.

12.7. Transaction: CreateAccount

Request: CreateRequest

Request: CreateRequest

Response: CreateResponse

Request creation of a new portal account.

Unlike a profile, a mesh account is specific to a particular Mesh portal. A mesh account must be created and accepted before a profile can be published.

12.7.1. Message: CreateRequest

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: MeshRequest

Inherits: MeshRequest

Account: String (Optional) Account identifier requested.

12.7.2. Message: CreateResponse

Inherits: MeshResponse

Reports the success or failure of a Create transaction.

[No fields]

12.8. Transaction: DeleteAccount

Request: DeleteRequest

Request: DeleteRequest

Response: DeleteResponse

Request deletion of a portal account.

Deletes a portal account but not the underlying profile. Once registered, profiles are permanent.

12.8.1. Message: DeleteRequest

Request deletion of a new portal account. The request specifies the requested account identifier.

Inherits: MeshRequest

Inherits: MeshRequest

Account: String (Optional) Account identifier to be deleted.

12.8.2. Message: DeleteResponse

Inherits: MeshResponse

Reports the success or failure of a Delete transaction.

[No fields]

12.9. Transaction: Get

Request: GetRequest

Request: GetRequest

Response: GetResponse

Search for data in the mesh that matches a set of properties described by a sequence of key/value pairs.

12.9.1. Message: GetRequest

Describes the Portal or Mesh data to be retreived.

Inherits: MeshRequest

Inherits: MeshRequest

Identifier: String (Optional) Lookup by profile ID

Account: String (Optional) Lookup by Account ID

KeyValues: KeyValue [0..Many] List of KeyValue pairs specifying the conditions to be met

SearchConstraints: SearchConstraints (Optional) Constrain the search to a specific time interval and/or limit the number and/or total size of data records returned.

Multiple: Boolean (Optional) If true return multiple responses if available

Full: Boolean (Optional) If true, the client requests that the full Mesh data record be returned containing both the Mesh entry itself and the Mesh metadata that allows the date and time of the publication of the Mesh entry to be verified.

12.9.2. Message: GetResponse

Reports the success or failure of a Get transaction. If a Mesh entry matching the specified profile is found, containsthe list of entries matching the request.

Inherits: MeshResponse

Inherits: MeshResponse

DataItems: DataItem [0..Many] List of mesh data records matching the request.

PageKey: String (Optional) If non-null, indicates that the number and/or size of the data records returned exceeds either the SearchConstraints specified in the request or internal server limits.

12.10. Transaction: Publish

Request: PublishRequest

Request: PublishRequest

Response: PublishResponse

Publish a profile or key escrow entry to the mesh.

12.10.1. Message: PublishRequest

Requests publication of the specified Mesh entry.

Inherits: MeshRequest

[No fields]

12.10.2. Message: PublishResponse

Reports the success or failure of a Publish transaction.

Inherits: MeshResponse

[No fields]

12.11. Transaction: Status

Request: StatusRequest

Request: StatusRequest

Response: StatusResponse

Request the current status of the mesh as seen by the portal to which it is directed.

The response to the status request contains the last signed checkpoint and proof chains for each of the peer portals that have been checkpointed.

[Not currently implemented]

12.11.1. Message: StatusRequest

Inherits: MeshRequest

Initiates a status transaction.

[No fields]

12.11.2. Message: StatusResponse

Reports the success or failure of a Status transaction.

Inherits: MeshResponse

Inherits: MeshResponse

LastWriteTime: DateTime (Optional) Time that the last write update was made to the Mesh

LastCheckpointTime: DateTime (Optional) Time that the last Mesh checkpoint was calculated.

NextCheckpointTime: DateTime (Optional) Time at which the next Mesh checkpoint should be calculated.

CheckpointValue: String (Optional) Last checkpoint value.

12.12. Transaction: ConnectStart

Request: ConnectStartRequest

Request: ConnectStartRequest

Response: ConnectStartResponse

Request connection of a new device to a mesh profile

12.12.1. Message: ConnectStartRequest

Inherits: MeshRequest

Initial device connection request.

SignedRequest: SignedConnectionRequest (Optional) Device connection request signed by thesignature key of the device requesting connection.

AccountID: String (Optional) Account identifier of account to which the device is requesting connection.

12.12.2. Message: ConnectStartResponse

Reports the success or failure of a ConnectStart transaction.

Inherits: MeshRequest

[No fields]

12.13. Transaction: ConnectStatus

Request: ConnectStatusRequest

Request: ConnectStatusRequest

Response: ConnectStatusResponse

Request status of pending connection request of a new device to a mesh profile

12.13.1. Message: ConnectStatusRequest

Inherits: MeshRequest

Request status information for a pending request posted previously.

AccountID: String (Optional) Account identifier for which pending connection information is requested.

DeviceID: String (Optional) Device identifier of device requesting status information.

12.13.2. Message: ConnectStatusResponse

Reports the success or failure of a ConnectStatus transaction.

Inherits: MeshRequest

Inherits: MeshRequest

Result: SignedConnectionResult (Optional) The signed ConnectionResult object.

12.14. Transaction: ConnectPending

Request: ConnectPendingRequest

Request: ConnectPendingRequest

Response: ConnectPendingResponse

Request a list of pending requests for an administration profile.

12.14.1. Message: ConnectPendingRequest

Inherits: MeshRequest

Specify the criteria for pending requests.

AccountID: String (Optional) The account identifier of the account for which pending connection requests are requested.

SearchConstraints: SearchConstraints (Optional) Constrain the search to a specific time interval and/or limit the number and/or total size of data records returned.

12.14.2. Message: ConnectPendingResponse

Reports the success or failure of a ConnectPending transaction.

Inherits: MeshRequest

Inherits: MeshRequest

Pending: SignedConnectionRequest [0..Many] A list of pending requests satisfying the criteria set out in the request.

PageKey: String (Optional) If non-null, indicates that the number and/or size of the data records returned exceeds either the SearchConstraints specified in the request or internal server limits.

12.15. Transaction: ConnectComplete

Request: ConnectCompleteRequest

Request: ConnectCompleteRequest

Response: ConnectCompleteResponse

Post response to a pending connection request.

12.15.1. Message: ConnectCompleteRequest

Reports the success or failure of a ConnectComplete transaction.

Inherits: MeshRequest

Inherits: MeshRequest

Result: SignedConnectionResult (Optional) The connection result to be posted to the portal. The result MUST be signed by a valid administration key for the Mesh profile.

AccountID: String (Optional) The account identifier to which the connection result is posted.

12.15.2. Message: ConnectCompleteResponse

Inherits: MeshRequest

Reports the success or failure of a ConnectComplete transaction.

[No fields]

12.16. Transaction: Transfer

Request: TransferRequest

Request: TransferRequest

Response: TransferResponse

Perform a bulk transfer of the log between the specified transaction identifiers. Requires appropriate authorization

[Not currently implemented]

12.16.1. Message: TransferRequest

Request a bulk transfer of the log between the specified transaction identifiers. Requires appropriate authorization

Inherits: MeshRequest

Inherits: MeshRequest

SearchConstraints: SearchConstraints (Optional) Constrain the search to a specific time interval and/or limit the number and/or total size of data records returned.

12.16.2. Message: TransferResponse

Inherits: MeshResponse

Reports the success or failure of a Transfer transaction. If successful, contains the list of Mesh records to be transferred.

DataItems: DataItem [0..Many] List of mesh data records matching the request.

PageKey: String (Optional) If non-null, indicates that the number and/or size of the data records returned exceeds either the SearchConstraints specified in the request or internal server limits.

13. Security Considerations

TBS

14. IANA Considerations

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

15. Acknowledgements

16. References

16.1. Normative References

[draft-hallambaker-mesh-architecture]
Hallam-Baker, P., "Mathematical Mesh: Architecture", draft-hallambaker-mesh-architecture-05 (work in progress), August 2018.

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

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011.

16.2. Informative References

[draft-hallambaker-mesh-developer]
Hallam-Baker, P., "Mathematical Mesh: Reference Implementation", draft-hallambaker-mesh-developer-07 (work in progress), April 2018.

16.3. URIs

[1] http://mathmesh.com/Documents/draft-hallambaker-mesh-reference.html

Author's Address

Phillip Hallam-Baker
Comodo Group Inc.

Email: philliph@comodo.com