Workgroup: Network Working Group
Internet-Draft: draft-hallambaker-mesh-schema
Published: 5 August 2021
Intended Status: Informational
Expires: 6 February 2022
Authors: P. M. Hallam-Baker
ThresholdSecrets.com

Mathematical Mesh 3.0 Part IV: Schema 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-schema.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 6 February 2022.

Table of Contents

1. Introduction
2. Definitions
  2.1. Requirements Language
  2.2. Defined Terms
  2.3. Related Specifications
  2.4. Implementation Status
3. Actors
  3.1. Accounts
  3.2. Device
    3.2.1. Activation
  3.3. Service
4. Catalogs
  4.1. Access
  4.2. Application
    4.2.1. Mesh Account
    4.2.2. SSH
    4.2.3. Mail
  4.3. Bookmark
  4.4. Contact
  4.5. Credential
  4.6. Device
  4.7. Network
  4.8. Publication
  4.9. Task
5. Spools
  5.1. Outbound
  5.2. Inbound
  5.3. Local
6. Cryptographic Operations
  6.1. Key Derivation from Seed
  6.2. Message Envelope and Response Identifiers.
  6.3. Proof of Knowledge of PIN
  6.4. EARL
  6.5. Key Agreement
  6.6. Service Cryptographic Operations
7. Mesh Assertions
  7.1. Encoding
  7.2. Mesh Profiles
  7.3. Mesh Connections

8. Architecture
  8.1. Mesh Account
    8.1.1. Account Profile
  8.2. Device Management
    8.2.1. The Device Catalog
    8.2.2. Mesh Devices
  8.3. Mesh Services
  8.4. Mesh Messaging
    8.4.1. Message Status
    8.4.2. Four Corner Model
    8.4.3. Traffic Analysis
9. Publications
  9.1. Contact Exchange
  9.2. Device Preconfiguration
  9.3. Device Description
10. Schema
  10.1. Shared Classes
    10.1.1. Classes describing keys
    10.1.2. Structure: KeyData
    10.1.3. Structure: CompositePrivate
  10.2. Assertion classes
    10.2.1. Structure: Assertion
    10.2.2. Structure: Condition
    10.2.3. Base Classes
    10.2.4. Structure: Connection
    10.2.5. Structure: Activation
    10.2.6. Structure: ActivationEntry
    10.2.7. Mesh Profile Classes
    10.2.8. Structure: Profile
    10.2.9. Structure: ProfileDevice
    10.2.10. Structure: ProfileAccount
    10.2.11. Structure: ProfileUser
    10.2.12. Structure: ProfileGroup
    10.2.13. Structure: ProfileService
    10.2.14. Structure: ProfileHost
    10.2.15. Connection Assertions
    10.2.16. Structure: ConnectionDevice
    10.2.17. Structure: ConnectionApplication
    10.2.18. Structure: ConnectionGroup
    10.2.19. Structure: ConnectionService
    10.2.20. Structure: ConnectionHost
    10.2.21. Activation Assertions
    10.2.22. Structure: ActivationDevice
    10.2.23. Structure: ActivationAccount
    10.2.24. Structure: ActivationApplication
  10.3. Data Structures
    10.3.1. Structure: Contact
    10.3.2. Structure: Anchor
    10.3.3. Structure: TaggedSource

10.3.4. Structure: ContactGroup
10.3.5. Structure: ContactPerson
10.3.6. Structure: ContactOrganization
10.3.7. Structure: OrganizationName
10.3.8. Structure: PersonName
10.3.9. Structure: NetworkAddress
10.3.10. Structure: NetworkProtocol
10.3.11. Structure: Role
10.3.12. Structure: Location
10.3.13. Structure: Bookmark
10.3.14. Structure: Reference
10.3.15. Structure: Task
10.4. Catalog Entries
10.4.1. Structure: CatalogedEntry
10.4.2. Structure: CatalogedDevice
10.4.3. Structure: CatalogedPublication
10.4.4. Structure: CatalogedCredential
10.4.5. Structure: CatalogedNetwork
10.4.6. Structure: CatalogedContact
10.4.7. Structure: CatalogedAccess
10.4.8. Structure: CryptographicCapability
10.4.9. Structure: CapabilityDecrypt
10.4.10. Structure: CapabilityDecryptPartial
10.4.11. Structure: CapabilityDecryptServiced
10.4.12. Structure: CapabilitySign
10.4.13. Structure: CapabilityKeyGenerate
10.4.14. Structure: CapabilityFairExchange
10.4.15. Structure: CatalogedBookmark
10.4.16. Structure: CatalogedTask
10.4.17. Structure: CatalogedApplication
10.4.18. Structure: CatalogedMember
10.4.19. Structure: CatalogedGroup
10.4.20. Structure: CatalogedApplicationSSH
10.4.21. Structure: CatalogedApplicationMail
10.4.22. Structure: CatalogedApplicationNetwork
10.5. Publications
10.5.1. Structure: DevicePreconfiguration
10.6. Messages
10.6.1. Structure: Message
10.6.2. Structure: MessageError
10.6.3. Structure: MessageComplete
10.6.4. Structure: MessagePinValidated
10.6.5. Structure: MessagePin
10.6.6. Structure: RequestConnection
10.6.7. Structure: AcknowledgeConnection
10.6.8. Structure: RespondConnection
10.6.9. Structure: MessageContact
10.6.10. Structure: GroupInvitation
10.6.11. Structure: RequestConfirmation

10.6.12. Structure: ResponseConfirmation
10.6.13. Structure: RequestTask
10.6.14. Structure: MessageClaim
10.6.15. Structure: ProcessResult
11. Security Considerations
12. IANA Considerations
13. Acknowledgements
14. Appendix A: Example Container Organization (not normative)
14.1. Device
14.1.1. Creating a new Mesh
14.1.2. Adding an Account
14.1.3. Adding a Device
14.2. Service
14.2.1. Creating a Service
14.2.2. Adding an Account
15. Appendix B: Collected Authentication and Encryption Requirements
15.1. Mesh Messaging
16. Normative References
17. Informative References

1. Introduction

This document describes the data structures of the Mathematical Mesh with illustrative examples. For an overview of the Mesh objectives and architecture, consult the accompanying *Architecture Guide* [draft-hallambaker-mesh-architecture]. For information on the implementation of the Mesh Service protocol, consult the accompanying *Protocol Reference* [draft-hallambaker-mesh-protocol]

This document has two main sections. The first section presents examples of the Mesh assertions, catalog entry and messages in use. The second section contains the schema reference. 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. Actors

The Mesh mediates interactions between three principal actors: **Accounts**, **Devices**, and **Services**.

Currently two account types are specified, **user accounts** which belong to an individual user and **group accounts** that are used to share access to confidential information between a group of users. It may prove useful to define new types of account over time or to eliminate the distinction entirely. When active a Mesh account is bound to a Mesh Service. The service to which an account is bound **MAY** be changed over time but an account can only be bound to a single service at a time.

A Mesh account is an abstract construct that (when active) is instantiated across one or more physical machines called a device. Each device that is connected to an account has a separate set of cryptographic keys that are used to interact with other devices connected to the account and **MAY** be provisioned with access to the account private keys which **MAY** or **MAY** NOT be mediated by the current Mesh Service.

A Mesh Service is an abstract construct that is provided by one or more physical machines called Hosts. A Mesh Host is a device that is attached to a service rather than an account.

3.1. Accounts

A Mesh Account is described by a Profile descended from Profile Account and contains a set of Mesh stores. Currently two account profiles are defined:

ProfileUser Describes a user account.

ProfileGroup Describes a group account used to share confidential information between a group of users.

Both types of profile specify the following fields:

ProfileSignature The public signature key used to authenticate the profile itself

AccountAddress The account name to which the account is currently bound. (e.g. alice@example.com, @alice).

ServiceUdf If the account is active, specifies the fingerprint of the service profile to which the account is currently bound.

AdministratorSignature The public signature key used to verify administrative actions on the account. In particular addition of devices to a user account or members to a group account.

AccountEncryption The public encryption key for the account. All messages sent to the account **MUST** be encrypted under this key. By definition, all data encrypted under this account is encrypted under this key.

User accounts specify two additional public keys, AccountSignature and AccountAuthentication which allow signature and authentication operations under the account context.

Every account contains a set of catalogs and spools that are managed by the service as directed by the contents of the associated Access catalog.

For example, the personal account profile Alice created is:

```
{
  "ProfileUser":{
    "ProfileSignature":{
      "Udf":"MC44-IZC3-IWZT-VCVZ-L2AG-HI4E-LOV2",
      "PublicParameters":{
        "PublicKeyECDH":{
          "crv":"Ed448",
          "Public":"W5ugEWyCz986Pw7xPP9-ap09PFpqfc9x_MB0Uq1brUpEg
  1ylIaitE5YaJIAeTb_Cc5y4Yr5D6kgA"}}},
    "AccountAddress":"alice@example.com",
    "ServiceUdf":"MBBY-E4VA-CMWF-52T7-ESLA-PGNT-CNNX",
    "AccountEncryption":{
      "Udf":"MCQX-PF7A-LY3G-3ZPI-AIYO-LR5D-YG6W",
      "PublicParameters":{
        "PublicKeyECDH":{
          "crv":"X448",
          "Public":"0ARE0VapsMqYf_P-tFnE0FZ2Zq2wn4e21viCH4Trhyg7Q
  0N8FXmbpYv_72_L2VKuliiiOhddT1OA"}}},
    "AdministratorSignature":{
      "Udf":"MCLJ-LCFX-ENE2-5NTF-GBNV-MW3P-VHSZ",
      "PublicParameters":{
        "PublicKeyECDH":{
          "crv":"Ed448",
          "Public":"GXmg8zk0pvV9b881JNvRm69dOTuL7oyYNwziQQltWF4aX
  ZMCTjf8EM3gtCU7wHF0zuFsd1WgX8GA"}}},
    "AccountAuthentication":{
      "Udf":"MDA5-PY6A-BG2L-7YN6-EAKP-O3QB-CG5T",
      "PublicParameters":{
        "PublicKeyECDH":{
          "crv":"X448",
          "Public":"XcHN7AbMTW1TMQOd0mXyAHG62O6j7YHgIGhp1Oaj5xTOK
  2uPzGxMeCeFOzX0f-YJXijL6hl_baWA"}}},
    "AccountSignature":{
      "Udf":"MCGB-4KCI-CP3G-FJSD-W7W2-UYKO-GNLU",
      "PublicParameters":{
        "PublicKeyECDH":{
          "crv":"Ed448",
          "Public":"cA1wcKtn1daj606UD0PJ8xLqFN0kOdxyjsF2QKrPZxdUY
  gSPRVRiLsLJOEl7cIsttpaCFwuPiDOA"}}}}}
```


3.2. Device

Every Mesh device has a set of private keys that are unique to that device. These keys **MAY** be installed during manufacture, installed from an external source after manufacture or generated on the device. If the platform capabilities allow, device private keys

SHOULD be bound to the device so that they cannot be extracted or exported without substantial effort.

The public keys corresponding to the device private keys are specified in a ProfileDevice. This **MUST** contain at least the following fields:

ProfileSignature The public signature key used to authenticate the profile itself.

BaseEncryption Public encryption key used as a share contribution to generation of device encryption keys to be used in the context of an account and to decrypt data during the process of connecting to an account.

BaseAuthentication Public authentication key used as a share contribution to generation of device authentication keys to be used in the context of an account and to authenticate the device to a service during the process of connecting to an account.

BaseSignature Public signature key used as a share contribution to generation of device authentication keys to be used in the context of an account.

For example, the device profile corresponding to Alice's coffee pot device is:

3.2.1. Activation

The device private keys are only used to perform cryptographic operations during the process of connecting a device to an account. During that connection process, a threshold key generation scheme is used to generate a second set of device keys bound to the account by combining the base key held by the device with a second device private key provided by the administration device approving the connection of the device to the account. The resulting key is referred to as the device key. The process of combining the base keys with the contributions to form the device keys is called Activation.

The activation record for Alice's coffee pot device is:

The Mesh protocols are designed so that there is never a need to export or escrow private keys of any type associated with a device, neither the base key, nor the device key nor the contribution from the administration device.

This approach to device configuration ensures that the keys that are used by the device when operating within the context of the account are entirely separate from those originally provided by the device

manufacturer or generated on the device, provided only that the key contributions from the administration device are sufficiently random and unguessable.

The public keys corresponding to the composite keys generated during the connection process are described in a ConnectionUser assertion signed by the administration key of the corresponding account.

The connection record for Alice's coffee pot device is:

The ConnectionUser assertion **MAY** be used in the same fashion as an X.509v3/PKIX certificate to mediate interactions between devices connected to the same account without the need for interaction with the Mesh service. Thus, a coffee pot device connected to the account can receive and authenticate instructions issued by a voice recognition device connected to that account.

While the ConnectionUser assertion **MAY** be used to mediate external interactions, this approach is typically undesirable as it provides the external parties with visibility to the internal configuration of the account, in particular which connected devices are being used on which occasions. Furthermore, the lack of the need to interact with the service means that the service is necessarily unable to mediate the exchange and enforce authorization policy on the interactions.

Device keys are intended to be used to secure communications between devices connected to the same account. All communication between Mesh accounts **SHOULD** be mediated by a Mesh service. This enables abuse mitigation by applying access control to every outbound and every inbound message.

Since Alice's coffee pot does not require the external communication right, the activation record for the coffee pot does not provide access to the account keys required to perform external communications. Alice's watch device does require access to the account keys so it can receive messages and task updates. But since it is a device that Alice has to carry on her person to use, it is a device that might easily be lost or stolen. Accordingly, the activation record for Alice's watch provides access to the account decryption and signature keys but in the form of threshold key shares mediated by the Mesh service. Thus, Alice's watch can sign and read message sent to the account but only under the control of the Mesh service.

3.3. Service

Mesh services are described by a ProfileService. This specifies the encryption, and signature authentication keys used to interact with the abstract service.

Since Mesh accounts and services are both abstract constructs, they cannot interact directly. A device connected to an account can only interact with a service by interacted with a device authorized to provide services on behalf of one or more accounts connected to the service. Such a device is called a Mesh Host.

Mesh hosts **MAY** be managed using the same ProfileDevice and device connection mechanism provided for management of user devices or by whatever other management protocols prove convenient. The only part of the Service/Host interaction that is visible to devices connected to a profile and to hosts connected to other services is the ConnectionHost structure that describes the set of device keys to use in interactions with that specific host.

4. Catalogs

Catalogs track sets of persistent objects associated with a Mesh Service Account. The Mesh Service has no access to the entries in any Mesh catalog except for the Device and Contacts catalog which are used in device authentication and authorization of inbound messages.

Each Mesh Catalog managed by a Mesh Account has a name of the form:

<prefix>_<name>

Where <prefix> is the IANA assigned service name. The assigned service name for the Mathematical Mesh is mmm. Thus, all catalogs specified by the Mesh schema have names prefixed with the sequence mmm_.

The following catalogs are currently specified within the Mathematical Mesh.

Access: mmm_Access Describes access control policy for performing operations on the account. The Access catalog is the only Mesh catalog whose contents are readable by the Mesh Service under normal circumstances.

Application: mmm_Application Describes configuration information for applications including mail (SMTP, IMAP, OpenPGP, S/MIME, etc) and SSH and for the MeshAccount application itself.

Bookmark: mmm_Bookmark Describes Web bookmarks and other citations allowing them to be shared between devices connected to the profile.

Contact: mmm_Contact Describes logical and physical contact information for people and organizations.

Credential: mmm_Credential
Describes credentials used to access network resources.

Device: mmm_Device Describes the set of devices connected to the account and the permissions assigned to them

Network: mmm_Network Describes network settings such as WiFi access points, IPSEC and TLS VPN configurations, etc.

Member: mmm_Member Describes the set of members connected to a group account.

Publication: mmm_Publication Describes data published under the account context. The data **MAY** be stored in the publication catalog itself or on a separate service (e.g. a Web server).

Task: mmm_CatalogTask Describes tasks assigned to the user including calendar entries and to do lists.

The Access, Publication, Device and Member catalogs are involved in Mesh Service Protocol interactions. These interactions are further described in the Protocol Reference [draft-hallambaker-mesh-protocol].

In many cases, the Mesh Catalog offers capabilities that represent a superset of the capabilities of an existing application. For example, the task catalog supports the appointment tracking functions of a traditional calendar application and the task tracking function of the traditional 'to do list' application. Combining these functions allows tasks to be triggered by other events other than the passage of time such as completion of other tasks, geographical presence, etc.

In such cases, the Mesh Catalog entries are designed to provide a superset of the data representation capabilities of the legacy formats and (where available) recent extensions. Where a catalog entry is derived from input presented in a legacy format, the original data representation **MAY** be attached verbatim to facilitate interoperability.

4.1. Access

The access catalog mmm_Access contains a list of access control entries granting a party authenticated using a particular cryptographic credential a specific privilege such as:

*Accept Mesh Messages of particular types

*Perform an operation on a private key known to the service.

As with the publication catalog, the access catalog provides information that is necessary for the Mesh Service to act on behalf of the user. It is therefore necessary to grant a decryption capability for this catalog during the process of binding the account to a service.

4.2. Application

The application catalog mmm_Application contains CatalogEntryApplication entries which describe the use of specific applications under the Mesh Service Account. Multiple application accounts for a single application **MAY** be connected to a single Mesh Service Account. Each account being specified in a separate entry.

The CatalogEntryApplication entries only contain configuration information for the application as it applies to the account as a whole. If the application requires separate configuration for individual devices, this is specified in separate activation records specified in the corresponding ConnectionDevice.

4.2.1. Mesh Account

Mesh Accounts are described by CatalogEntryAccount entries. The corresponding activation records for the connected devices contain the contributions used to derive the private keys for use of the account.

The CatalogEntryAccount entry is described in the section describing Mesh accounts above.

4.2.2. SSH

SSH configuration profiles are described by CatalogEntryApplicationSSH entries. The corresponding activation records for the connected devices contain the contributions used to derive the private keys.

A user may have separate SSH configurations for separate purposes within a single Mesh Account. This allows a system administrator servicing multiple clients to maintain separate SSH profiles for each of her customers allowing credentials to be easily (and verifiably) revoked at contract termination.

The SSH profile contains the information that is stored in the known_hosts and authorized_keys files of SSH clients and servers.

4.2.3. Mail

Mail configuration profiles are described by one or more CatalogEntryApplicationMail entries, one for each email account

connected to the Mesh profile. The corresponding activation records for the connected devices contain information used to provide the device with the necessary decryption information.

Entries specify the email account address(es), the inbound and outbound server configuration and the cryptographic keys to be used for S/MIME and OpenPGP encryption.

4.3. Bookmark

The bookmark catalog mmm_bookmark contains CatalogEntryBookmark entries which describe Web bookmarks and other citations allowing them to be shared between devices connected to the profile.

The fields currently supported by the Bookmarks catalog are currently limited to the fields required for tracking Web bookmarks. Specification of additional fields to track full academic citations is a work in progress.

```
{
  "CatalogedBookmark":{
    "Uri":"http://www.site1.com",
    "Title":"site1",
    "Path":"Sites.1"}}
```

4.4. Contact

The contact catalog mmm_contact contains CatalogEntryContact entries which describe

```
{
  "CatalogedContact":{
    "Key":"MC44-IZC3-IWZT-VCVZ-L2AG-HI4E-LOV2",
    "Self":true,
    "Contact":{
      "ContactPerson":{
        "Id":"MC44-IZC3-IWZT-VCVZ-L2AG-HI4E-LOV2",
        "Anchors":[{
            "Udf":"MC44-IZC3-IWZT-VCVZ-L2AG-HI4E-LOV2",
            "Validation":"Self"}
          ],
        "NetworkAddresses":[{
            "Address":"alice@example.com",
            "EnvelopedProfileAccount":[{
                "EnvelopeId":"MC44-IZC3-IWZT-VCVZ-L2AG-HI4E-LOV2",
                "dig":"S512",
                "ContentMetaData":"ewogICJVbmlxdWVJZCI6ICJNQzQ0LU
  laQzMtSVdaVC1WQ1ZaLUwyQUctSEk0RS1MT1YyIiwKICAiTWVzc2FnZVR5cGUiOiA
  iUHJvZmlsZVVzZXIiLAogICJjdHkiOiAiYXBwbGljYXRpb24vbW1tL29iamVjdCIs
  CiAgIkNyZWF0ZWQiOiAiMjAyMS0wOC0wNVQxNjozNzozMVoifQ"},
              "ewogICJQcm9maWxlVXNlciI6IHsKICAgICJQcm9maWxlU2lnbm
  F0dXJlIjogewogICAgICAiVWRmIjogIk1DNDQtSVpDMy1JV1pULVZDVlotTDJBRy1
  ISTRFLUxPVjIiLAogICAgICAiUHVibGljUGFyYW1ldGVycyI6IHsKICAgICAgICAi
  UHVibGljS2V5RUNESCI6IHsKICAgICAgICAgICJjcnYiOiAiRWQ0NDgiLAogICAgI
  CAgICAgIlB1YmxpYyI6ICJXNXVnRVd5Q3o5ODZQdzd4UFA5LWFwMDlQRnBxZmM5eF
  9NQjBVcTFiclVwRWcxeWxJYWl0CiAgRTVZYUpJQWVUYl9DYzV5NFlyNUQ2a2dBIn1
  9fSwKICAgICJBY2NvdW50QWRkcmVzcyI6ICJhbGljZUBleGFtcGxlLmNvbSIsCiAg
  ICAiU2VydmljZVVkZiI6ICJNQkJZLUU0VkEtQ01XRi01MlQ3LUVTTEEtUEdOVC1DT
  k5YIiwKICAgICJBY2NvdW50RW5jcnlwdGlvbiI6IHsKICAgICAgICJlVkZiI6ICJNQ1
  FYLVBGN0EtTFkzRy0zWlBJLUFJWU8tTFI1RC1ZRzZXIiwKICAgICAgICJlB1YmxpY1B
  hcmFtZXRlcnMiOiB7CiAgICAgICAgICJlB1YmxpY0tleUVDREgiOiB7CiAgICAgICAg
  ICAiY3J2IjogIlg0NDgiLAogICAgICAgICAgICJlB1YmxpYyI6ICIwQVJFMFZhcHNNc
  VlmX1AtdEZuRTBGWjJacTJ3bjRlMjF2aUNINFRyaHlnN1EwTjhGWG1iCiAgcFl2Xz
  cyX0wyVkt1bGlpaU9oZGRUMU9BIn19fSwKICAgICJBZG1pbmlzdHJhdG9yU2lnbmF
  0dXJlIjogewogICAgICAiVWRmIjogIk1DTEotTENGWC1FTkUyLTVOVEYtR0JOVi1N
  VzNQLVZIU1oiLAogICAgICAiUHVibGljUGFyYW1ldGVycyI6IHsKICAgICAgICAiU
  HVibGljS2V5RUNESCI6IHsKICAgICAgICAgICJjcnYiOiAiRWQ0NDgiLAogICAgIC
  AgICAgICAgIlB1YmxpYyI6ICJHWG1nOHprMHB2VjliODgxSk52Um02OWRPVHVMN295WU5
  3emlRUWx0V0Y0YVhaTUNUamY4CiAgRU0zZ3RDVTd3SEYwenVGc2QxV2dYOEdBIn19
  fSwKICAgICJBY2NvdW50QXV0aGVudGljYXRpb24iOiB7CiAgICAgICJVZGYiOiAiT
  URBNS1QWTZBLUJHMkwtN1lONi1FQUtQLU8zUUItQ0c1VCIsCiAgICAgICJQdWJsaW
  NQYXJhbWV0ZXJzIjogewogICAgICAgICJQdWJsaWNLZXlFQ0RIIjogewogICAgICA
  gICAgImNydiI6ICJYNDQ4IiwKICAgICAgICAgICJQdWJsaWMiOiAiWGNITjdBYk1U
  VzFUTVFPZDBtWHlBSEc2Mk82ajdZSGdJR2hwMU9hajV4VE9LMnVQekd4TQogIGVDZ
  UZPelgwZi1ZSlhpakw2aGxfYmFXQSJ9fX0sCiAgICAiQWNjb3VudFNpZ25hdHVyZS
  I6IHsKICAgICAgIlVkZiI6ICJNQ0dCLTRLQ0ktQ1AzRy1GSlNELVc3VzItVVlLTy1
  HTkxVIiwKICAgICAgIlB1YmxpY1BhcmFtZXRlcnMiOiB7CiAgICAgICAgICJlB1Ymxp
  Y0tleUVDREgiOiB7CiAgICAgICAgICAgICAiY3J2IjogIkVkNDQ4IiwKICAgICAgICAgI
  CJQdWJsaWMiOiAiY0Exd2NLdG4xZGFqNjA2VUQwUEo4eExxRk4wa09keHlqc0YyUU
```

```
tyUFp4ZFVZZ1NQUlZSaQogIExzTEpPRWw3Y0lzdHRwYUNGd3VQaURPQSJ9fX19fQ",
            {
              "signatures":[{
                  "alg":"S512",
                  "kid":"MC44-IZC3-IWZT-VCVZ-L2AG-HI4E-LOV2",
                  "signature":"p1UxtlxOGp8wd7oESFQkmqR6uV9s-cW4
WtonkwN1BbH9F1KN6lllLPhnNBhlmstQ06cm7BEYtR6AwG2f1yOM1IjdPFzLmm53M
BA-g4GLebflsg87h_kvT9JVSnHq7MDY6ewMo97Boay7r26qf-Ci4xIA"}
                ],
              "PayloadDigest":"gNprAKlPBfHdWXEdD7djEQ8IoJzHAJS-
zBlZW6Bj8xjHqsbEhqbSO4AkIEizON5HsGbMSu_BQl_NPSrKZWS9pw"}
            ],
          "Protocols":[{
              "Protocol":"mmm"}
            ]}
        ],
      "Sources":[{
          "Validation":"Self",
          "EnvelopedSource":[{
              "dig":"S512",
              "ContentMetaData":"ewogICJNZXNzYWdlVHlwZSI6ICJDb2
50YWN0UGVyc29uIiwKICAiY3R5IjogImFwcGxpY2F0aW9uL21tbS9vYmplY3QiLAo
gICJDcmVhdGVkIjogIjIwMjEtMDgtMDVUMTY6Mzc6MzFaIn0"},
            "ewogICJDb250YWN0UGVyc29uIjogewogICAgIkFuY2hvcnMiOi
BbewogICAgICAgICJVZGYiOiAiTUM0NC1JWkMzLUlXWlQtVkNWWi1MMkFHLUhJNEU
tTE9WMiIsCiAgICAgICAgIlZhbGlkYXRpb24iOiAiU2VsZiJ9XSwKICAgICJOZXR3
b3JrQWRkcmVzc2VzIjogW3sKICAgICAgICAiQWRkcmVzcyI6ICJhbGljZUBleGFtc
GxlLmNvbSIsCiAgICAgICAgIkVudmVsb3BlZFByb2ZpbGVBY2NvdW50IjogW3sKIC
AgICAgICAgICAgIkVudmVsb3BlSWQiOiAiTUM0NC1JWkMzLUlXWlQtVkNWWi1MMkF
HLUhJNEUtTE9WMiIsCiAgICAgICAgICAgICJkaWciOiAiUzUxMiIsCiAgICAgICAg
ICAgICJDb250ZW50TWV0YURhdGEiOiAiZXdvZ0lDSlZibWx4ZFdWSlpDSTZJQ0pOU
XpRMExVbGFRek10U1ZkYVZDMQogIFdRMVphTFV3eVFVY3RTRWswUlMxTVQxWXlJaX
dLSUNBaVRXVnpjMkZuWlZSNWNHVWlPaUFpVUhKdlptbHNaCiAgVlZ6WlhJaUxBb2d
JQ0pqZEhraU9pQWlZWEJ3YkdsallYUnBiMjR2YlcxdEwyOWlhbVZqZENJc0NpQWdJ
a04KICB5WldGMFpXUWlPaUFpTWpBeU1TMHdPQzB3TlZReE5qb3pOem96TVZvaWZRI
n0sCiAgICAgICAgICAiZXdvZ0lDSlFjbTltYVd4bFZYTmxjaUk2SUhzS0lDQWdJQ0
pRY205bWFXeAogIGxVMmxuYm1GMGRYSmxJam9nZXdvZ0lDQWdJQ0FpVldSbUlqb2d
JazFETkRRdFNWcERNeTFKVjFwVUxWWkRWCiAgbG90VERKQlJ5MUlTVFJGTFV4UFZq
SWlMQW9nSUNBZ0lDQWlVSFZpYkdsalVHRnlZVzFsZEdWeWN5STZJSHMKICBLSUNBZ
0lDQWdJQ0FpVUhWaWJHbGpTMlY1UlVORVNDSTZJSHNLSUNBZ0lDQWdJQ0FnSUNKam
NuWWlPaUFpUgogIFdRME5EZ2lMQW9nSUNBZ0lDQWdJQ0FnSWxCMVlteHBZeUk2SUN
KWE5YVm5SVmQ1UTNvNU9EWlFkemQ0VUZBCiAgNUxXRndNRGxRUm5CeFptTTVlRjlO
UWpCVmNURmljbFZ3UldjeGVXeEpZV2wwQ2lBZ1JUVlpZVXBKUVdWVVkKICBsOURZe
lY1TkZseU5VUTJhMmRCSW4xOWZTd0tJQ0FnSUNKQlkyTnZkVzUwUVdSa2NtVnpjeU
k2SUNKaGJHbAogIGpaVUJsZUdGdGNHeGxMbU52YlNJc0NpQWdJQ0FpVTJWeWRtbGp
aVlZrWmlJNklDSk5Ra0paTFVVMFZrRXRRCiAgMDFYUmkwMU1sUTNMVVZUVEVFdFVF
ZE9WQzFEVGs1WUlpd0tJQ0FnSUNKQlkyTnZkVzUwUlc1amNubHdkR2wKICB2YmlJN
klIc0tJQ0FnSUNBZ0lsVmtaaUk2SUNKTlExRllMVkJHTjBFdFRGa3pSeTB6V2xCSk
xVRkpXVTh0VAogIEZJMVJDMVpSelpYSWl3S0lDQWdJQ0FnSWxCMVlteHBZMUJoY21
```

```
GdFpYUmxjbk1pT2lCN0NpQWdJQ0FnSUNBCiAgZ0lsQjFZbXhwWTB0bGVVVkRSRWdp
T2lCN0NpQWdJQ0FnSUNBZ0lDQWlZM0oySWpvZ0lsZzBORGdpTEFvZ0kKICBDQWdJQ
0FnSUNBZ0lsQjFZbXhwWXlJNklDSXdRVkpGTUZaaGNITk5jVmxtWDFBdGRFWnVSVE
JHV2pKYWNUSgogIDNialJsTWpGMmFVTklORlJ5YUhsbk4xRXdUamhHV0cxaUNpQWd
jRmwyWHpjeVgwd3lWa3QxYkdscGFVOW9aCiAgR1JVTVU5QkluMTlmU3dLSUNBZ0lD
SkJaRzFwYm1semRISmhkRzl5VTJsbmJtRjBkWEpsSWpvZ2V3b2dJQ0EKICBnSUNBa
VZXUm1Jam9nSWsxRFRFb3RURU5HV0MxRlRrVXlMVFZPVkVZdFIwSk9WaTFOVnpOUU
xWWklVMW9pTAogIEFvZ0lDQWdJQ0FpVUhWaWJHbGpVR0Z5WVcxbGRHVnljeUk2SUh
zS0lDQWdJQ0FnSUNBaVVIVmliR2xqUzJWCiAgNVJVTkVTQ0k2SUhzS0lDQWdJQ0Fn
SUNBZ0lDSmpjbllpT2lBaVJXUTBORGdpTEFvZ0lDQWdJQ0FnSUNBZ0kKICBsQjFZb
XhwWXlJNklDSkhXRzFuT0hwck1IQjJWamxpT0RneFNrNTJVbTAyT1dSUFZIVk1OMj
k1V1U1M2VtbAogIFJVV3gwVjBZMFlWaGFUVU5VYW1ZNENpQWdSVTB6WjNSRFZUZDN
TRVl3ZW5WR2MyUXhWMmRZT0VkQkluMTlmCiAgU3dLSUNBZ0lDSkJZMk52ZFc1MFFY
VjBhR1Z1ZEdsallYUnBiMjRpT2lCN0NpQWdJQ0FnSUNKVlpHWWlPaUEKICBpVFVSQ
k5TMVFXVFpCTFVKSE1rd3ROMWxPTmkxRlFVdFFMVTh6VVVJdFEwYzFWQ0lzQ2lBZ0
lDQWdJQ0pRZAogIFdKc2FXTlFZWEpoYldWMFpYSnpJam9nZXdvZ0lDQWdJQ0FnSUN
KUWRXSnNhV05MWlhsRlEwUklJam9nZXdvCiAgZ0lDQWdJQ0FnSUNBZ0ltTnlkaUk2
SUNKWU5EUTRJaXdLSUNBZ0lDQWdJQ0FnSUNKUWRXSnNhV01pT2lBaVcKICBHTklUa
mRCWWsxVVZ6RlVUVkZQWkRCdFdIbEJTRWMyTWs4MmFqZFpTR2RKUjJod01VOWhhal
Y0VkU5TE1uVgogIFFla2Q0VFFvZ0lHVkRaVVpQZWxnd1ppMVpTbGhwYWt3MmFHeGZ
ZbUZYUVNKOWZYMHNDaUFnSUNBaVFXTmpiCiAgM1Z1ZEZOcFoyNWhkSFZ5WlNJNklI
c0tJQ0FnSUNBZ0lsVmtaaUk2SUNKTlEwZENMVFJMUTBrdFExQXpSeTEKICBHU2xOR
UxWYzNWekl0VlZsTFR5MUhUa3hWSWl3S0lDQWdJQ0FnSWxCMVlteHBZMUJoY21GdF
pYUmxjbk1pTwogIGlCN0NpQWdJQ0FnSUNBZ0lsQjFZbXhwWTB0bGVVVkRSRWdpT2l
CN0NpQWdJQ0FnSUNBZ0lDQWlZM0oySWpvCiAgZ0lrVmtORFE0SWl3S0lDQWdJQ0Fn
SUNBZ0lDSlFkV0pzYVdNaU9pQWlZMEV4ZDJOTGRHNHhaR0ZxTmpBMlYKICBVUXdVR
W80ZUV4eFJrNHdhMDlrZUhscWMwWXlVVXR5VUZwNFpGVlpaMU5RVWxaU2FRb2dJRX
h6VEVwUFJXdwogIDNZMGx6ZEhSd1lVTkdkM1ZRYVVSUFFTSjlmWDE5ZlEiLAogICA
gICAgICAgewogICAgICAgICAgICAic2lnbmF0dXJlcyI6IFt7CiAgICAgICAgICAg
ICAgICAiYWxnIjogIlM1MTIiLAogICAgICAgICAgICAgICAgImtpZCI6ICJNQzQ0L
UlaQzMtSVdaVC1WQ1ZaLUwyQUctSEk0RS1MT1YyIiwKICAgICAgICAgICAgICAgIC
JzaWduYXR1cmUiOiAicDFVeHRseE9HcDh3ZDdvRVNGUWttcVI2dVY5cy1jVzRXdG9
ua3dOMUJiSDlGMUtONgogIGxsbExQaG5OQmhsbXN0UTA2Y203QkVZdFI2QXdHMmYx
eU9NMUlqZFBGekxtbTUzTUJBLWc0R0xlYmZsc2c4CiAgN2hfa3ZUOUpWU25IcTdNR
Fk2ZXdNbzk3Qm9heTdyMjZxZi1DaTR4SUEifV0sCiAgICAgICAgICAgICAgICJQYXlsb2
FkRGlnZXN0IjogImdOcHJBS2xQQmZIZFdYRWREN2RqRVE4SW9KekhBSlMtekJsWlc
2Qmo4eGpIcQogIHNiRWhxYlNPNEFrSUVpek9ONUhzR2JNU3VfQlFsX05QU3JLWldT
OXB3In1dLAogICAgICAgICAgICAgICJQcm90b2NvbHMiOiBbewogICAgICAgICAgICAiUHJvd
G9jb2wiOiAibW1tIn1dfV19fQ",
            {
              "signatures":[{
                  "alg":"S512",
                  "kid":"MCGB-4KCI-CP3G-FJSD-W7W2-UYKO-GNLU",
                  "signature":"JzkD8nCuWRjhfEQegsmuuV8iIhOjZ7jK
uke3tNmKS2-zg1w_7b8kUZdaAKn3KTm-5fsUZrFq13eAej7CnvXgRT7j58zhHCHm1
UKpxb13CQwFyNo7rCT31HKeOgS0V8Qa1npKMK6ZzsupGUttvuJ1qxIA"}
                ],
              "PayloadDigest":"lZVGf5kCDcfU544bSV1GOy3F3VjR7mv7
avq71IF1Lqsmd5A4A02yBbGOdzXJt_gOWbFXIi1NzhAKFpUFAvjiaw"}
```

```
            ]}
          ]}}}}
```

The fields of the contact catalog provide a superset of the capabilities of vCard [RFC2426].

The Contact catalog is typically used by the MeshService as a source of authorization information to perform access control on inbound and outbound message requests. For this reason, Mesh Service **SHOULD** be granted read access to the contacts catalog by providing a decryption entry for the service.

4.5. Credential

The credential catalog mmm_credential contains CatalogEntryCredential entries which describe credentials used to access network resources.

```
{
  "CatalogedCredential":{
    "Service":"ftp.example.com",
    "Username":"alice1",
    "Password":"password"}}
```

Only username/password credentials are stored in the credential catalog. If public key credentials are to be used, these **SHOULD** be managed as an application profile allowing separate credentials to be created for each device.

4.6. Device

The device catalog mmm_Device contains CatalogEntryDevice entries which describe the devices connected to the account and the permissions assigned to them.

Each device connected to a Mesh Account has an associated CatalogEntryDevice entry that includes the activation and connection records for the account. These records are described in further detail in section REF _Ref54628559 \r \h 0.

4.7. Network

The network catalog contains CatalogEntryNetwork entries which describe network settings, IPSEC and TLS VPN configurations, etc.

```
{
  "CatalogedNetwork":{
    "Service":"myWiFi",
    "Password":"securePassword"}}
```

4.8. Publication

The publication catalog mmm_Publication contains CatalogEntryPublication entries which describe content published through the account.

4.9. Task

The Task catalog mmm_Task contains CatalogEntryTask entries which describe tasks assigned to the user including calendar entries and to do lists.

The fields of the task catalog currently reflect those offered by the iCalendar specification [RFC5545]. Specification of additional fields to allow task triggering on geographic location and/or completion of other tasks is a work in progress.

```
{
  "CatalogedTask":{
    "Title":"SomeItem",
    "Key":"NC72-I2UG-KRON-M4MI-TCCX-5N7X-KRZ4"}}
```

5. Spools

Spools are DARE Containers containing an append only list of messages sent or received by an account. Three spools are currently defined:

Inbound Messages sent to the account. These are encrypted under the account encryption keys of the sender and receiver that were current at the time the message was sent.

Outbound Messages sent from the account. These are encrypted under the account encryption keys of the sender and receiver that were current at the time the message was sent.

Local Messages sent from the account for internal use. These are encrypted under the encryption key of the intended recipient alone. This is either the account administration encryption key or a device encryption key.

Every Mesh Message has a unique message identifier. Messages created at the beginning of a new messaging protocol interaction are assigned a random message identifier. Responses to previous messages are assigned message identifiers formed from the message identifier to which they respond by means of a message digest function.

Every Mesh Message stored in a spool is encapsulated in an envelope which bears a unique identifier that is formed by applying a message digest function to the message identifier. Each stored message has an associated state which is initially set to the state Initial and **MAY** be subsequently altered by one or more MessageComplete messages subsequently appended to the spool. The allowable message states depending upon the spool in question.

5.1. Outbound

The outbound spool stores messages that are to be or have been sent and MessageComplete messages reporting changes to the status of the messages stored on the spool.

Messages posted to the outbound spool have the state Initial, Sent, Received or Refused:

Initial The initial state of a message posted to the spool.

Sent The Mesh Service of the sender has delivered the message to the Mesh Service of the recipient which accepted it.

Received The Mesh Service of the sender has delivered the message to the Mesh Service of the recipient and the recipient has acknowledged receipt.

Refused The Mesh Service of the sender has delivered the message to the Mesh Service of the recipient which refused to accept it.

MessageComplete messages are only valid when posted to the spool by the service.

5.2. Inbound

The inbound spool stores messages that have been received by the Mesh service servicing the account and MessageComplete messages reporting changes to the status of the messages stored on the spool.

Messages posted to the outbound spool have the state Initial, Read:

Initial The initial state of a message posted to the spool.

Read The message has been read.

A message previously marked as read **MAY** be returned to the unread state by marking it as being in the Initial state.

5.3. Local

The local spool stores messages that are used for administrative functions. In normal circumstances, only administrator devices and the Mesh Service require access to the local spool.

The local spool is used to store MessagePin messages used to notify administration devices that a PIN code has been registered for some purpose and RespondConnection messages used to inform a device of the result of a connection request.

The local spool is used in a device connection operation to provide a device with the activation and connection records required to access the service as an authorized client. Servicing these requests requires that the service be able to access messages stored in the spool by envelope id.

Messages posted to the outbound spool have the states Initial, Closed:

Initial The initial state of a message posted to the spool.

Closed The action associated with the message has been completed.

6. Cryptographic Operations

The Mesh makes use of various cryptographic operations including threshold operations. For convenience, these are gathered here and specified as functions that are referenced by other parts of the specification.

6.1. Key Derivation from Seed

Mesh Keys that derived from a seed value use the mechanism described in [draft-hallambaker-mesh-udf]. Use of the keyname parameter allows multiple keys for different uses to be derived from a single key. Thus escrow of a single seed value permits recovery of all the private keys associated with the profile.

The keyname parameter is a string formed by concatenating identifiers specifying the key type, the actor that will use the key and the key operation:

6.2. Message Envelope and Response Identifiers.

Every Mesh message has a unique Message Identifier MessageId. The MakeID() function is used to calculate the value of Envelope

Identifier and Response identifier from the message identifier as follows:

```
static string MakeID(string udf, string content) {
    var (code, bds) = UDF.Parse(udf);
    return code switch
        {
            UdfTypeIdentifier.Digest_SHA_3_512 =>
                UDF.ContentDigestOfDataString(
                bds, content, cryptoAlgorithmId:
                    CryptoAlgorithmId.SHA_3_512),
            _ => UDF.ContentDigestOfDataString(
            bds, content, cryptoAlgorithmId:
                    CryptoAlgorithmId.SHA_2_512),
            };
```

Where the values of content are given as follows:

String String

For example:

```
MessageID
    = NBF4-4PTH-UTSC-RYHZ-EQMK-BVH7-MU3F

EnvelopeID
    = MD22-62AQ-PVVT-NPUS-ATUC-MQH4-PWV2

ResponseID
    = MCZ5-YD47-KD23-MHLY-F34Z-PFVI-47VK
```

6.3. Proof of Knowledge of PIN

Mesh Message classes that are subclasses of MessagePinValidated **MAY** be authenticated by means of a PIN. Currently two such messages are defined: MessageContact used in contact exchange and RequestConnection message used in device connection.

The PIN codes used to authenticate MessagePinValidated messages are UDF Authenticator strings. The type code of the identifier specifies the algorithm to be used to authenticate the PIN code and the Binary Data Sequence value specifies the key.

The inputs to the PIN proof of knowledge functions are:

PIN: string

A UDF Authenticator. The type code of the identifier specifies the algorithm to be used to authenticate the PIN code and the Binary Data Sequence value specifies the key.

Action: string A code determining the specific action that the PIN code **MAY** be used to authenticate. By convention this is the name of the Mesh message type used to perform the action.

Account: string The account for which the PIN code is issued.

ClientNonce: binary Nonce value generated by the client using the PIN code to authenticate its message.

PayloadDigest: binary The PayloadDigest of a DARE Envelope that contains the message to be authenticated. Note that if the envelope is encrypted, this value is calculated over the ciphertext and does not provide proof of knowledge of the plaintext.

The following values of Action are currently defined:

String String

These inputs are used to derive values as follows:

```
alg =           UdfAlg (PIN)
pinData =       UdfBDS (PIN)
saltedPINData = MAC (Action, pinData)
saltedPIN =     UDFPresent (HMAC_SHA_2_512 + saltedPINData)
PinId =         UDFPresent (MAC (Account, saltedPINData))
```

The issuer of the PIN code stores the value saltedPIN for retrieval using the key PinId.

The witness value for a Dare Envelope with payload digest PayloadDigest authenticated by a PIN code whose salted value is saltedPINData, issued by account Account is given by PinWitness() as follows:

```
witnessData =   Account.ToUTF8() + ClientNonce + PayloadDigest
witnessValue =  MAC (witnessData , saltedPINData)
```

For example, to generate saltedPIN for the pin AAJP-KPYV-NSB4-GIYL-AM used to authenticate a an action of type Device:

```
pin = AAJP-KPYV-NSB4-GIYL-AM
action = message.

alg = UdfAlg (PIN)
    = Authenticator_HMAC_SHA_2_512

hashalg = default (alg, HMAC_SHA_2_512)

pinData = UdfBDS (PIN)
    = System.Byte[]

saltedPINData
    = hashalg(pinData, hashalg);
    = System.Byte[]

saltedPIN = UDFPresent (hashalg + saltedPINData)
    = ADBB-JTKR-RGGL-ACEM-TW6E-ZFY6-NQXF
```

The PinId binding the pin to the account alice@example.com is

```
Account =  alice@example.com

PinId = UDFPresent (MAC (Account, saltedPINData))
    = ADVR-47UA-ZHDX-6CFS-GDDD-RYTA-LSB6
```

Where MAC(data, key) is the message authentication code algorithm specified by the value of alg.

When an administrative device issues a PIN code, a Message PIN is appended to the local spool. This has the MessageId PinId and specifies the value saltedPIN in the field of that name.

When PIN code authentication is used, a message of type MessagePinValidated specifies the values ClientNonce, PinWitness and PinId in the fields of those names. These values are used to authenticate the inner message data specified by the AuthenticatedData field.

6.4. EARL

The UDF Encrypted Authenticated Resource Locator mechanism is used to publish data and provide means of authentication and access through a static identifier such as a QR code.

This mechanism is used to allow contact exchange by means of a QR code printed on a business card and to connect a device to an

account using a static identifier printed on the device in the form of a QR code.

In both cases, the information is passed using the EARL format described in [draft-hallambaker-mesh-udf].

6.5. Key Agreement

All Mesh Protocol requests except for the HelloRequest and every response **MUST** be authenticated under the device key of the host or device making the request.

Initial authentication is achieved by performing a Key agreement under the DeviceAuthentication key of each of the hosts and combining the result with nonce values provided by the requestor and respondent using a KDF function as follows:

Two bindings are currently planned.

DARE Envelope over HTTPS The request or response is encapsulated in a DARE Envelope that is exchanged by means of a HTTP POST method over a TLS transport. The shared secret is used as the key on Message Authentication Code that authenticates the request payload.

UDP Transport Presents the same information as for the DARE Envelope over HTTPS case but in a compact encoding using the shared secret and an authenticated encryption scheme to pass the required information.

Once authentication has been performed, the same pair of devices **MAY** re-authenticate using the previously agreed key. To facilitate stateless implementation, a host specifies an opaque identifier to be used to identify the shared secret on subsequent uses which **MAY** be used to recover the shared secret from the opaque identifier.

[To be specified]

6.6. Service Cryptographic Operations

A Mesh Service acts as the counterparty for threshold operations allowing mitigation of the risk of compromise of an individual device connected to a user account or an insider threat from an individual member of a group account.

When acting in this role, the Mesh service controls the use of the cryptographic function but does not have the ability to perform the

action either by itself or by collaborating with other services to which the account has been bound in the past.

Note that this approach limits rather than eliminates trust in the service. As with services presenting themselves as 'zero trust', a Mesh service becomes a trusted service after a sufficient number of breaches in other parts of the system have occurred. And the user trusts the service to provide availability of the service.

Three service cryptographic operations are currently specified:

Threshold Key Share A private key share s, held by the service is split into key shares x, y such that $a = x + y$. One key share is encrypted under a decryption key held by the service. The other is encrypted under a public key specified by the party making the request.

Threshold Key Agreement A private key share s, held by the service is used to calculate the value $(sl + c).P$ where l, c are integers specified by the requestor and P is a point on the curve.

Threshold Signature A private key share s, held by the service is used to calculate a contribution to a threshold signature scheme.

The implementation of the cryptographic operations described above is described in [draft-hallambaker-threshold] and [draft-hallambaker-threshold-sigs].

7. Mesh Assertions

Mesh Assertions are signed DARE Envelopes that contain one of more claims. Mesh Assertions provide the basis for trust in the Mathematical Mesh.

Mesh Assertions are divided into two classes. Mesh Profiles are self-signed assertions. Assertions that are not self-signed are called declarations. The only type of declaration currently defined is a Connection Declaration describing the connection of a device to an account.

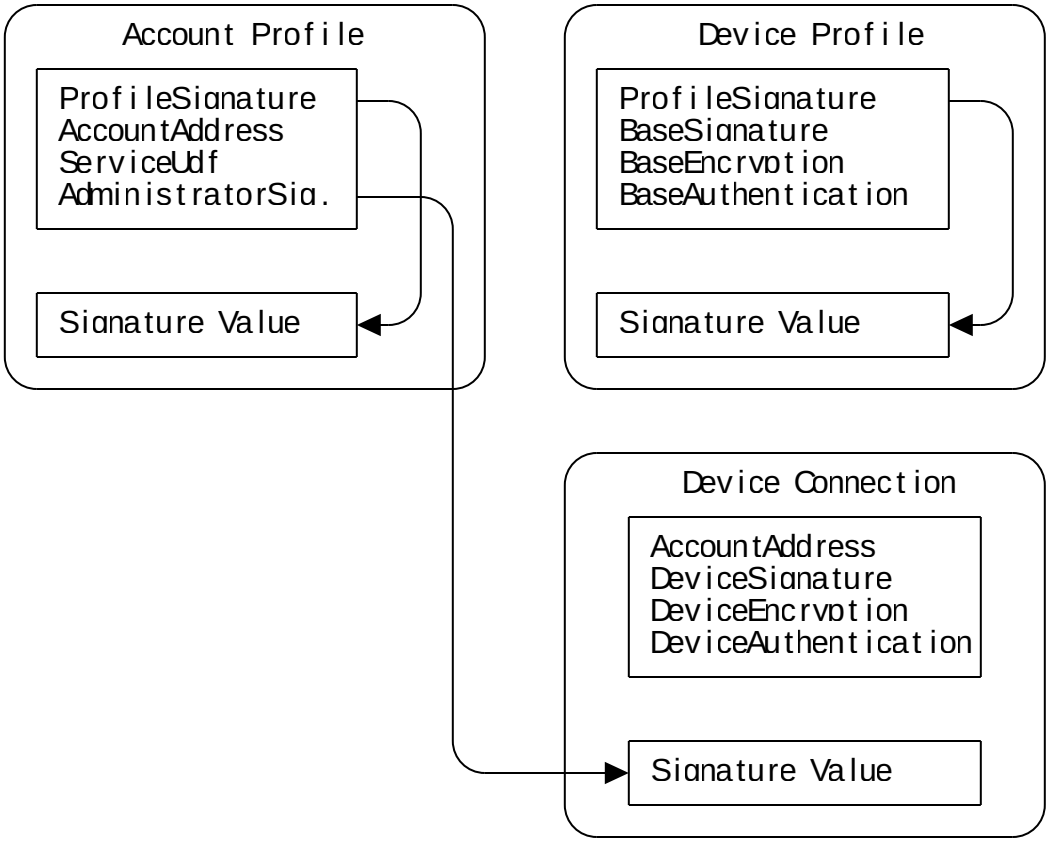

Figure 1: Profiles And Connections

7.1. Encoding

The payload of a Mesh Assertion is a JSON encoded object that is a subclass of the Assertion class which defines the following fields:

Identifier An identifier for the assertion.

Updated The date and time at which the assertion was issued or last updated

NotaryToken An assertion may optionally contain one or more notary tokens issued by a Mesh Notary service. These establish a proof that the assertion was signed after the date the notary token was created.

Conditions A list of conditions that **MAY** be used to verify the status of the assertion if the relying party requires.

The implementation of the NotaryToken and Conditions mechanisms is to be specified in [draft-hallambaker-mesh-notary] at a future date.

Note that the implementation of Conditions differs significantly from that of SAML. Relying parties are required to process condition

clauses in a SAML assertion to determine validity. Mesh Relying parties **MAY** verify the conditions clauses or rely on the trustworthiness of the provider.

The reason for weakening the processing of conditions clauses in the Mesh is that it is only ever possible to validate a conditions clause of any type relative to a ground truth. In SAML applications, the relying party almost invariably has access to an independent source of ground truth. A Mesh device connected to a Mesh Service does not. Thus the types of verification that can be achieved in practice are limited to verifying the consistency of current and previous statements from the Mesh Service.

7.2. Mesh Profiles

Mesh Profiles perform a similar role to X.509v3 certificates but with important differences:

* Profiles describe credentials, they do not make identity statements

* Profiles do not expire, there is therefore no need to support renewal processing.

* Profiles may be modified over time, the current and past status of a profile being recorded in an append only log.

Profiles provide the axioms of trust for the Mesh PKI. Unlike in the PKIX model in which all trust flows from axioms of trust held by a small number of Certificate Authorities, every part in the Mesh contributes their own axiom of trust.

It should be noted however that the role of Certificate Authorities is redefined rather than eliminated. Rather than making assertions whose subject is represented by identities which are inherently mutable and subjective, Certificate Authorities can now make assertions about immutable cryptographic keys.

Every Profile **MUST** contain a SignatureKey field and **MUST** be signed by the key specified in that field.

A Profile is valid if and only if:

* There is a SignatureKey field.

* The profile is signed under the key specified in the SignatureKey field.

A profile has the status current if and only if:

- The Profile is valid
- Every Conditions clause in the profile is understood by the relying party and evaluates to true.

7.3. Mesh Connections

A Mesh connection is an assertion describing the connection of a device or a member to an account.

Mesh connections provide similar functionality to 'end-entity' certificates in PKIX but with the important proviso that they are only used to provide trust between a device connected to an account and the service to which that account is bound and between the devices connected to an account.

A connection is valid with respect to an account with profile *P* if and only if:

- The profile *P* is valid
- The AuthorityUdf field of the connection is consistent with the UDF of *P*
- The profile is signed under the key specified in the AdministrationKey field of *P*.
- Any conditions specified in the profile are met

A connection has the status current with respect to an account with profile if and only if:

- The connection is valid with respect to the account with profile *P*.
- The profile P is current.

A device is authenticated with respect to an account with profile P if and only if:

- The connection is valid with respect to the account with profile *P*.
- The device has presented an appropriate proof of knowledge of the DeviceAuthentication key specified in the connection.

8. Architecture

The Mesh architecture has four principal components:

Mesh Account A collection of information (contacts, calendar entries, inbound and outbound messages, etc.) belonging to a user who uses the Mesh to management.

Mesh Device Management The various functions that manage binding of devices to a Mesh to grant access to information and services bound to that account.

Mesh Service Provides network services through which devices and other Mesh users may interact with a Mesh Account.

Mesh Messaging An end to end secure messaging service that allows short messages (less than 32KB) to be exchanged between Mesh Accounts and between the Mesh devices connected to a particular account.

The separation of accounts and services as separate components is a key distinction between the Mesh and earlier Internet applications. A Mesh account belongs to the owner of the Mesh and not the Mesh Service Provider which the user may change at any time of their choosing.

A Mesh Account May be active or inactive. By definition, an active Mesh account is serviced by exactly one Mesh Service, an inactive Mesh account is not serviced by a Mesh Service. A Mesh Service Provider **MAY** offer a backup service for accounts hosted by other providers. In this case the backup provider is connected to the account as a Mesh device, thus allowing the backup provider to maintain a copy of the stores contained in the account and facilitating a rapid transfer of responsibility for servicing the account should that be desired. The use of backup providers is described further in [draft-hallambaker-mesh-discovery].

8.1. Mesh Account

Mesh Accounts contains all the stateful information (contacts, calendar entries, inbound and outbound messages, etc.) related to a particular persona used by the owner.

By definition a Mesh Account is active if it is serviced by a Mesh Service and inactive otherwise. A Mesh user **MAY** change their service provider at any time. An active Mesh Account is serviced by exactly one Mesh Service at once but a user **MAY** register a 'backup' service provider to their account in the same manner as adding an advice. This ensures that the backup service is pre-populated with all the

information required to allow the user to switch to the new provider without interruption of service.

Each Mesh account is described by an Account Profile. Currently separate profile Account Profile are defined for user accounts and group accounts. It is not clear if this distinction is a useful one.

8.1.1. Account Profile

A Mesh account profile provides the axiom of trust for a mesh user. It contains a Master Signature Key and one or more Administration Signature Keys. The unique identifier of the master profile is the UDF of the Master Signature Key.

An Account Profile **MUST** specify an EscrowEncryption key. This key **MAY** be used to escrow private keys used for encryption of stored data. They **SHOULD NOT** be used to escrow authentication keys and **MUST NOT** be used to escrow signature keys.

A user should not need to replace their account profile unless they intend to establish a separate identity. To minimize the risk of disclosure, the Profile Signature Key is only ever used to sign updates to the account profile itself. This allows the user to secure their Profile Signature Key by either keeping it on hardware token or device dedicated to that purpose or by using the escrow mechanism and paper recovery keys as described in this document.

8.1.1.1. Creating a ProfileMaster

Creating a ProfileMaster comprises the steps of:

0. Creating a Master Signature key.

1. Creating an Online Signing Key

2. Signing the ProfileMaster using the Master Signature Key

3. Persisting the ProfileMaster on the administration device to the CatalogHost.

4. (Optional) Connecting at least one Administration Device and granting it the ActivationAdministration activation.

8.1.1.2. Updating a ProfileMaster

Updating a ProfileMaster comprises the steps of:

0. Making the necessary changes.

1. Signing the ProfileMaster using the Master Signature Key

2. Persisting the ProfileMaster on the administration device to the CatalogHost.

8.2. Device Management

Device management allows a collection of devices belonging to a user to function as a single personal Mesh.

The device management functions are principally concerned with the catalog containing the entries describing the connected devices.

8.2.1. The Device Catalog

Each Mesh Account has a Device Catalog CatalogDevice associated with it. The Device Catalog is used to manage the connection of devices to the Personal Mesh and has a CatalogEntryDevice for each device currently connected to the catalog.

Each Administration Device **MUST** have access to an up-to-date copy of the Device Catalog in order to manage the devices connected to the Mesh. The Mesh Service protocol **MAY** be used to synchronize the Device Catalog between administration devices in the case that there is more than one administration device.

The CatalogEntryDevice contains fields for the device profile, device private and device connection.

8.2.2. Mesh Devices

The principle of radical distrust requires us to consider the possibility that a device might be compromised during manufacture. Once consequence of this possibility is that when an administration device connects a new device to a user's personal Mesh, we cannot put our full trust in either the device being connected or the administration device connecting it.

This concern is resolved by (at minimum) combining keying material generated from both sources to create the keys to be used in the context of the user's personal Mesh with the process being fully verified by both parties.

Additional keying material sources could be added if protection against the possibility of compromise at both devices was required but this is not supported by the current specifications.

A device profile provides the axiom of trust and the key contributions of the device. When bound to an account, the base keys specified in the Device Profile are combined with the key data provided in the Activation device to construct the keys the device will use in the context of the account.

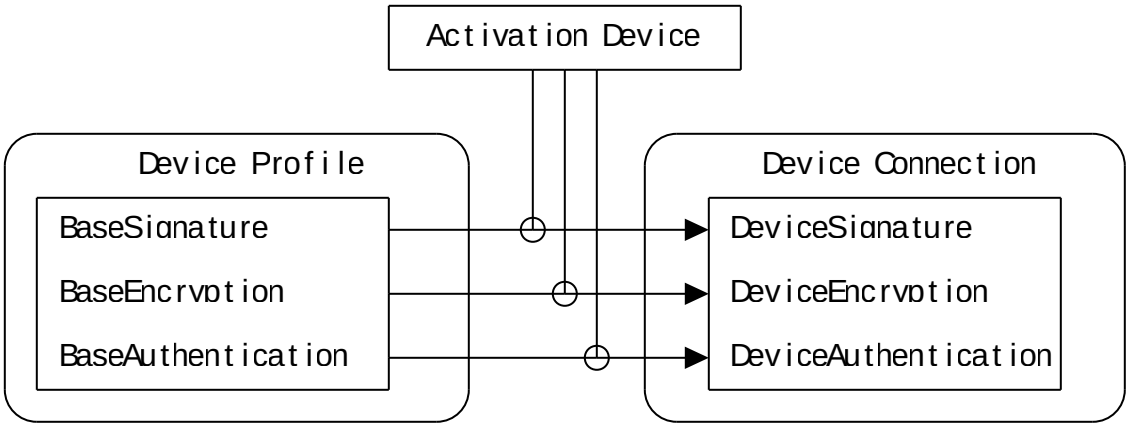

Figure 2: Mapping of Device Profile and Device Private to Device Connection Keys.

Unless exceptional circumstances require, a device should not require more than one Device profile even if the device supports use by multiple users under different accounts. But a device **MAY** have multiple profiles if this approach is more convenient for implementation.

The derivation of the Connection encryption and signature keys from the Profile and Private contributions in this example is shown in [draft-hallambaker-mesh-cryptography].

8.2.2.1. Creating a ProfileDevice

Creating a ProfileDevice comprises the steps of:

0. Creating the necessary key

1. Signing the ProfileDevice using the Master Signature Key

2. Once created, a ProfileDevice is never changed. In the unlikely event that any modification is required, a completely new ProfileDevice **MUST** be created.

8.2.2.2. Connection to a Personal Mesh

Devices are only connected to a personal Mesh by an administration device. This comprises the steps of:

0. Generating the PrivateDevice keys.

1. Creating the ConnectionDevice data from the public components of the ProfileDevice and PrivateDevice keys and signing it using the administration key.

2. Creating the Activations for the device and signing them using the administration key.

3. Creating the CatalogEntryDevice for the device and adding it to the CatalogDevice of the Personal Mesh.

4. If the Personal Mesh has accounts that are connected to a Mesh Service, synchronizing the CatalogEntryDevice to those services.

These steps are usually performed through use of the Mesh Protocol Connection mechanism. However, Mesh clients **MAY** support additional mechanisms as circumstances require provided that the appropriate authentication and private key protection controls are provided.

8.3. Mesh Services

A Mesh Service provides one or more Mesh Hosts that support Mesh Accounts through the Mesh Web Service Protocol.

Mesh Services and Hosts are described by Service Profiles and Host Profiles. The means by which services manage the hosts through which they provide service is outside the scope of this document.

As with a Device connected to a Mesh Account, a the binding of a Host to the service it supports is described by a connection record:

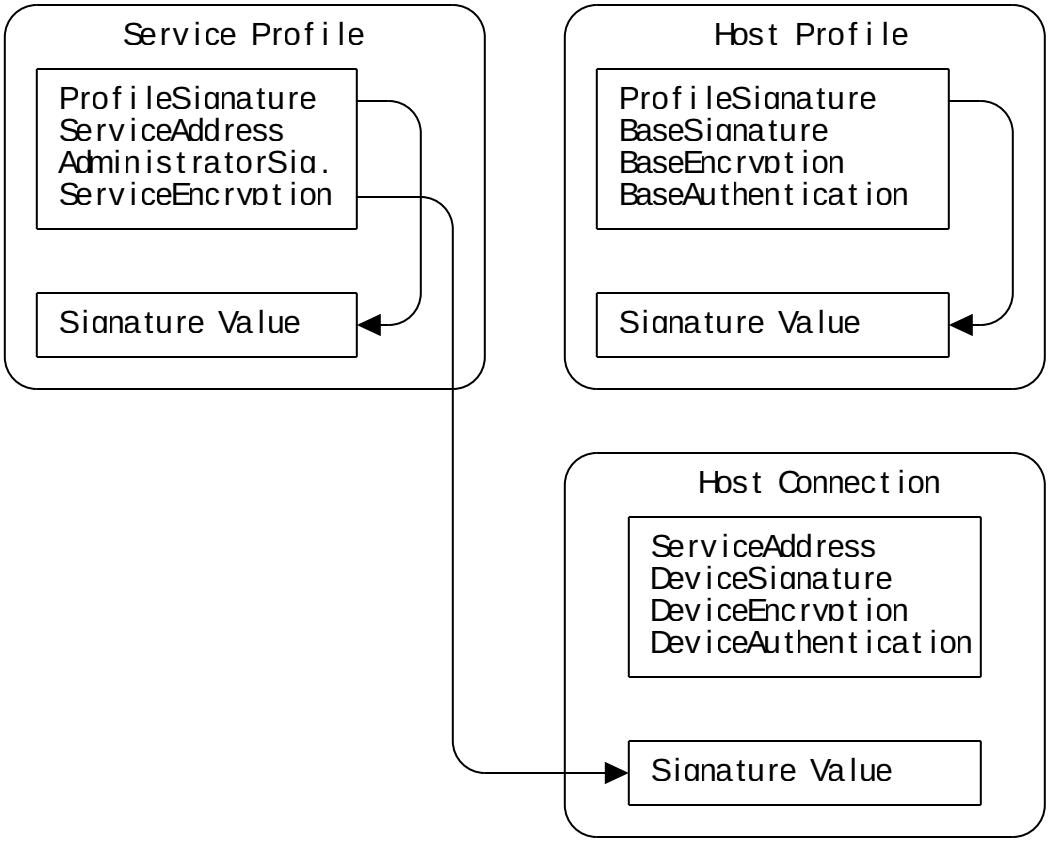

Figure 3: Service Profile and Delegated Host Assertion.

The credentials provided by the ProfileService and ProfileHost are distinct from those provided by the WebPKI that typically services TLS requests. WebPKI credentials provide service introduction and authentication while a Mesh ProfileHost only provides authentication.

Unless exceptional circumstances require, a service should not need to revise its Service Profile unless it is intended to change its identity. Service Profiles **MAY** be countersigned by Trusted Third Parties to establish accountability.

8.4. Mesh Messaging

Mesh Messaging is an end-to-end secure messaging system used to exchange short (32KB) messages between Mesh devices and services. In cases where exchange of longer messages is required, Mesh Messaging **MAY** be used to provide a control plane to advise the intended message recipient(s) of the type of data being offered and the means of retrieval (e.g an EARL).

All communications between Mesh accounts takes the form of a Mesh Message carried in a Dare Envelope. Mesh Messages are stored in two

spools associated with the account, the SpoolOutbound and the SpoolInbound containing the messages sent and received respectively.

This document only describes the representation of the messages within the message spool. The Mesh Service protocol by which the messages are exchanged between devices and services and between services is described in [draft-hallambaker-mesh-protocol].

8.4.1. Message Status

As previously described in section ###, every message stored in a spool has a specified state. The range of allowable states is defined by the message type. New message states **MAY** be defined for new message types as they are defined.

By default, messages are appended to a spool in the Initial state, but a spool entry **MAY** specify any state that is valid for that message type.

The state of a message is changed by appending a completion message to the spool as described in [draft-hallambaker-mesh-protocol].

Services **MAY** erase or redact messages in accordance with local site policy. Since messages are not removed from the spool on being marked deleted, they may be undeleted by marking them as read or unread. Marking a message deleted **MAY** make it more likely that the message will be removed if the sequence is subsequently purged.

8.4.2. Four Corner Model

A four-corner messaging model is enforced. Mesh Services only accept outbound messages from devices connected to accounts that it services. Inbound messages are only accepted from other Mesh Services. This model enables access control at both the outbound and inbound services

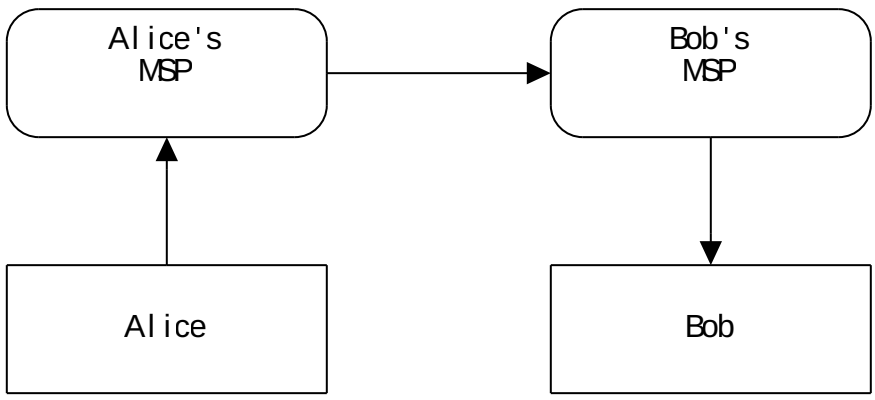

Figure 4: Four Corner Messaging Model

The outbound Mesh Service checks to see that the request to send a message does not violate its acceptable use policy. Accounts that make a large number of message requests that result in complaints **SHOULD** be subject to consequences ranging from restriction of the number and type of messages sent to suspending or terminating messaging privileges. Services that fail to implement appropriate controls are likely to be subject to sanctions from either their users or from other services.

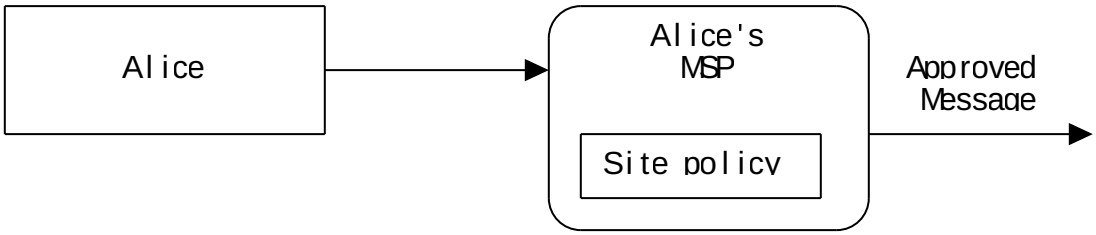

Figure 5: Performing Access Control on Outbound Messages

The inbound Mesh Service also checks to see that messages received are consistent with the service Acceptable Use Policy and the user's personal access control settings.

Mesh Services that fail to police abuse by their account holders **SHOULD** be subject to consequences in the same fashion as account holders.

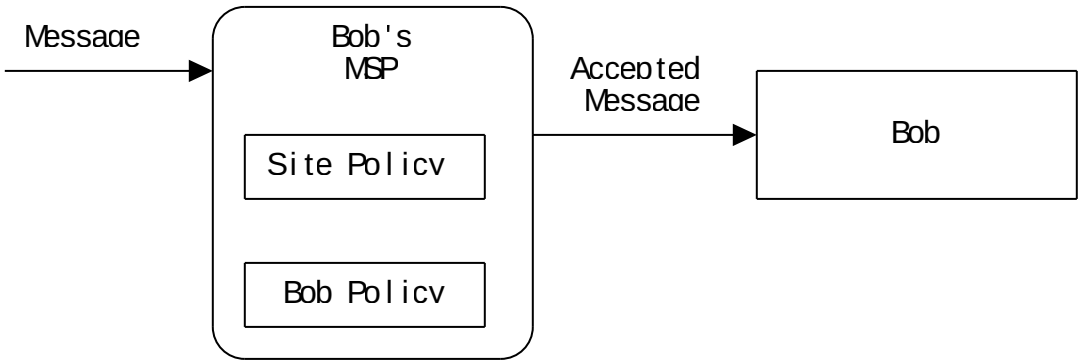

Figure 6: Performing Access Control on Inbound Messages

8.4.3. Traffic Analysis

The Mesh Messaging protocol as currently specified provides only limited protection against traffic analysis attacks. The use of TLS

to encrypt communication between Mesh Services limits the effectiveness of na?ve traffic analysis mechanisms but does not prevent timing attacks unless dummy traffic is introduced to obfuscate traffic flows.

The limitation of the message size is in part intended to facilitate use of mechanisms capable of providing high levels of traffic analysis such as mixmaster and onion routing but the current Mesh Service Protocol does not provide support for such approaches and there are no immediate plans to do so.

9. Publications

Static QR codes **MAY** be used to allow contact exchange or device connection. In either case, the QR code contains an EARL providing the means of locating, decrypting and authenticating the published data.

The use of EARLs as a means of publishing encrypted data and the use of EARLs for location, decryption and authentication is discussed in [draft-hallambaker-mesh-dare] .

9.1. Contact Exchange

When used for contact exchange, the envelope payload is a CatalogedContact record.

Besides allowing for exchange of contact information on a business card, a user might have their contact information printed on personal property to facilitate return of lost property.

9.2. Device Preconfiguration

The static QR code device connection interaction allows a device with no keyboard, display or other user affordances to be connected to a Mesh account.

The information necessary to establish communication with the device and to complete a device connection workflow is provided by means of a DevicePreconfiguration record accessed by means of an EARL.

For example, Alice's coffee pot was preconfigured for connection to a Mesh account at the factory and the following DevicePreconfiguration record created:

```
{
  "DevicePreconfiguration":{
    "EnvelopedProfileDevice":[{
        "EnvelopeId":"MC6H-2IFS-4OXA-U2YQ-QZ6Y-RZWX-JC3Q",
        "dig":"S512",
        "ContentMetaData":"ewogICJVbmlxdWVJZCI6ICJNQzZILTJJRlMtNE
  9YQS1VMllRLVFaNlktUlpXWC1KQzNRIiwKICAiTWVzc2FnZVR5cGUiOiAiUHJvZml
  sZURldmljZSIsCiAgImN0eSI6ICJhcHBsaWNhdGlvbi9tbW0vb2JqZWN0IiwKICAi
  Q3JlYXRlZCI6ICIyMDIxLTA4LTA1VDE2OjM3OjQzWiJ9"},
      "ewogICJQcm9maWxlRGV2aWNlIjogewogICAgIlByb2ZpbGVTaWduYXR1cm
  UiOiB7CiAgICAgICJVZGYiOiAiTUM2SC0ySUZTLTRPWEEtVTJZUS1RWjZZLVJaV1g
  tSkMzUSIsCiAgICAgICJQdWJsaWNQYXJhbWV0ZXJzIjogewogICAgICAgICJQdWJs
  aWNLZXlFQ0RIIjogewogICAgICAgICAgImNydiI6ICJFZDQ0OCIsCiAgICAgICAgI
  CAiUHVibGljIjogIjdaV0RBem50bXl4aFV2elhDdmw5dEF4UlhlQnoyV3V3VHFsTl
  Zub2pzb0s3NTNfTnZqOVoKICBYV19KT0p6UkFlU082U1lyQ3VYNThlVUEifX19LAo
  gICAgIkVuY3J5cHRpb24iOiB7CiAgICAgICJVZGYiOiAiTUFOVy1BN1pFLVJVQ04t
  R0wyQi03VEtYLVYyVUYtS1RHMiIsCiAgICAgICJQdWJsaWNQYXJhbWV0ZXJzIjoge
  wogICAgICAgICJQdWJsaWNLZXlFQ0RIIjogewogICAgICAgICAgImNydiI6ICJYND
  Q4IiwKICAgICAgICAgICJQdWJsaWMiOiAiejZKdUtyTFVfaG5najBwdzF4eHVGcDZ
  uS3ZVSTYtZVVzOS1ZR3dtYlplMHhFR2FOM2hMZwogIDZORkdHTFFDSmpfc2kyWmhw
  UGhOWTJZQSJ9fX0sCiAgICAiU2lnbmF0dXJlIjogewogICAgICAiVWRmIjogIk1CV
  VEtSFVNWi1IWk9SLUVBQUEtRERCSC1VTUc1LVQzNksiLAogICAgICAiUHVibGljUG
  FyYW1ldGVycyI6IHsKICAgICAgICAiUHVibGljS2V5RUNESCI6IHsKICAgICAgICA
  gICJjcnYiOiAiRWQ0NDgiLAogICAgICAgICAgIlB1YmxpYyI6ICJpaFgwQ3B6d1Zx
  RFU5NWgybWlPbkswYWtjUkFFSEgxUHc3cGJPdEpldkVhM3dDaHg0VjZMCiAgenZme
  kw4Smw2SHVlMXhlOXpBNDBPY0dBIn19fSwKICAgICAgICJBdXRoZW50aWNhdGlvbiI6IH
  sKICAgICAgICAgIlVkZiI6ICJNQTZKLTNURk8tQ09YVi1UUE5GLUNPQUYtRExaTy1MN09
  aIiwKICAgICAgICAgICAgIlB1YmxpY1BhcmFtZXRlcnMiOiB7CiAgICAgICAgICAgICAgIlB1YmxpY0tl
  eUVDREgiOiB7CiAgICAgICAgICAgICAgICAiY3J2IjogIlg0NDgiLAogICAgICAgICAgICAgICAgIlB1Y
  mxpYyI6ICIzVGZDTWxpdk5Ta1k0b0UtdkczMVUzbXZ2M1YzWTVZUlg5YzhrSmdaOE
  czRDB6eFhjR1pDCiAgTWZueE5KTUItYTNKbktSRF9DZHZvRGFBIn19fX19",
      {
        "signatures":[{
            "alg":"S512",
            "kid":"MC6H-2IFS-4OXA-U2YQ-QZ6Y-RZWX-JC3Q",
            "signature":"TbXfDJHac9bh1mRbgm6pcCOfjAC4kAe9xb-Bf2XG
  V3idDknG7xcFsvv-hOCGnNd0ca3k35tcBXiAiMMs5xfysFT_5sOpbWNsivC3o63yW
  AN4332vd-2O1-VuRRE-j4F9cR2kn9PPW8oOkBcdSOUY8SAA"}
          ],
        "PayloadDigest":"wCvIv_8ngV6HWhb3MG-sEidG4X7Xsvl8NnJf9BJO
  XNKVymd5rr6_TwzAGThCLw4-kh45Pb5OA7gYJZh-zTojVA"}
      ],
    "EnvelopedConnectionDevice":[{
        "dig":"S512",
        "ContentMetaData":"ewogICJNZXNzYWdlVHlwZSI6ICJDb25uZWN0aW
  9uRGV2aWNlIiwKICAiY3R5IjogImFwcGxpY2F0aW9uL21tbS9vYmplY3QiLAogICJ
  DcmVhdGVkIjogIjIwMjEtMDgtMDVUMTY6Mzc6NDNaIn0"},
      "ewogICJDb25uZWN0aW9uRGV2aWNlIjogewogICAgIkF1dGhlbnRpY2F0aW
  9uIjogewogICAgICAiVWRmIjogIk1BTlctQTdaRS1SVUNOLUdMMkItN1RLWC1WMlV
```

```
  GLUtURzIiLAogICAgICAiUHVibGljUGFyYW1ldGVycyI6IHsKICAgICAgICAiUHVi
  bGljS2V5RUNESCI6IHsKICAgICAgICAgICJjcnYiOiAiWDQ0OCIsCiAgICAgICAgI
  CAiUHVibGljIjogIno2SnVLckxVX2huZ2owcHcxeHh1RnA2bkt2VUk2LWVVczktWU
  d3bWJaZTB4RUdhTjNoTGcKICA2TkZHR0xRQ0pqX3NpMlpocFBoTlkyWUEifX19LAo
  gICAgIlNpZ25hdHVyZSI6IHsKICAgICAgIlVkZiI6ICJNQlVRLUhVTVotSFpPUi1F
  QUFBLUREQkgtVU1HNS1UMzZLIiwKICAgICAgIlB1YmxpY1BhcmFtZXRlcnMiOiB7C
  iAgICAgICAgIlB1YmxpY0tleUVDREgiOiB7CiAgICAgICAgICAiY3J2IjogIkVkND
  Q4IiwKICAgICAgICAgICJQdWJsaWMiOiAiaWhYMENwendWcURVOTVoMm1pT25LMGF
  rY1JBRUhIMVB3N3BiT3RKZXZFYTN3Q2h4NFY2TAogIHp2ZnpMOEpsNkh1ZTF4ZTl6
  QTQwT2NHQSJ9fX0sCiAgICAiRW5jcnlwdGlvbiI6IHsKICAgICAgIlVkZiI6ICJNQ
  U5XLUE3WkUtUlVDTi1HTDJCLTdUS1gtVjJVRi1LVEcyIiwKICAgICAgIlB1YmxpY1
  BhcmFtZXRlcnMiOiB7CiAgICAgICAgIlB1YmxpY0tleUVDREgiOiB7CiAgICAgICA
  gICAiY3J2IjogIlg0NDgiLAogICAgICAgICAgIlB1YmxpYyI6ICJ6Nkp1S3JMVV9o
  bmdqMHB3MXh4dUZwNm5LdlVJNi1lVXM5LVlHd21iWmUweEVHYU4zaExnCiAgNk5GR
  0dMUUNKal9zaTJaaHBQaE5ZMllBIn19fX19",
      {
        "signatures":[{
            "alg":"S512",
            "kid":"MB4T-TWDN-IFT2-PU43-N4CQ-YTOK-AOIO",
            "signature":"BlY5NDotpRXQhVyURm-IS1aldtt5B9xZXO1bQ9CX
  qKstEgRG1EVPTf9K5WamWo3cToRMxMNg832AyEbp4nOmnfwkRsNACc9wElgEuGCPh
  2IPKBO6FpysIxZSSEUDV7Q08zmyfoRO9lxFMdavQ56RnDYA"}
          ],
        "PayloadDigest":"2U9aYu58E5UvYkBQAL5eNMoK-PhNBeGg3xCG_swk
  CdYSkOh2bznyoh7BF_JVFz01lJWd9VkIYGYej8IEBFgqJA"}
      ],
    "PrivateKey":{
      "PrivateKeyUDF":{
        "PrivateValue":"ZAAQ-AO3M-SYLX-X7ET-MZHC-BURC-V2OE-ZWUX-ZWP
F-KAL4-6C3G-6IPV-2D76-MT4N",
        "KeyType":"MeshProfileDevice"}},
    "ConnectUri":"mcu://maker@example.com/EALU-LAUT-XB5I-I2ZV-M2VN-
NKAW-QLYX-JTS7-64DV-44NF-7YS7-2SHB-7ZE7-4"}}
```


To connect to the coffee pot, Alice first scans the QR code with her administrative device which uses the PIN code and service to retrieve, decrypt and authenticate the DevicePreconfiguration record. Future versions of the specification will allow this record to specify means by which the administration device can establish direct peer-to-peer communication to complete the connection process by any communication modality supported by both devices (e.g. IR, Bluetooth, WiFi-Direct, etc.)

The use of the publication mechanism in device connection is discussed further in [draft-hallambaker-mesh-protocol].

9.3. Device Description

The device description publication is a JSON Record that describes a device that is available for connection.

[Not yet implemented.]

10. Schema

10.1. Shared Classes

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

10.1.1. Classes describing keys

10.1.2. Structure: KeyData

The KeyData 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.

NotBefore: DateTime (Optional) If present specifies a time instant that use of the private key is not valid before.

NotOnOrAfter: DateTime (Optional) If present specifies a time instant that use of the private key is not valid on or after.

10.1.3. Structure: CompositePrivate

Inherits: Key UDF fingerprint of the bound device key (if used).
DeviceKeyUdf: String (Optional)

10.2. Assertion classes

Classes that are derived from an assertion.

10.2.1. Structure: Assertion

Parent class from which all assertion classes 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.

10.2.2. Structure: Condition

Parent class from which all condition classes are derived.

[No fields]

10.2.3. Base Classes

Abstract classes from which the Profile, Activation and Connection classes are derrived.

10.2.4. Structure: Connection

Inherits: Assertion UDF of the connection target.
SubjectUdf: String (Optional)
AuthorityUdf: String (Optional) UDF of the connection source.

10.2.5. Structure: Activation

Inherits: Assertion
Contains the private activation information for a Mesh application running on a specific device

ActivationKey: String (Optional) Secret seed used to derive keys that are not explicitly specified.

Entries: ActivationEntry [0..Many] Activation of named resources.

10.2.6. Structure: ActivationEntry

Resource: String (Optional) Name of the activated resource

Key: KeyData (Optional) The activation key or key share

10.2.7. Mesh Profile Classes

Classes describing Mesh Profiles. All Profiles are Assertions derrived from Assertion.

10.2.8. Structure: Profile

Inherits: Assertion

Parent class from which all profile classes are derived

ProfileSignature: KeyData (Optional) The permanent signature key used to sign the profile itself. The UDF of the key is used as the permanent object identifier of the profile. Thus, by definition, the KeySignature value of a Profile does not change under any circumstance.

10.2.9. Structure: ProfileDevice

Inherits: Profile

Describes a mesh device.

Description: String (Optional) Description of the device

BaseEncryption: KeyData (Optional) Base key contribution for encryption keys. Also used to decrypt activation data sent to the device during connection to an account.

BaseAuthentication: KeyData (Optional) Base key contribution for authentication keys. Also used to authenticate the device during connection to an account.

BaseSignature: KeyData (Optional) Base key contribution for signature keys.

10.2.10. Structure: ProfileAccount

Base class for the account profiles ProfileUser and ProfileGroup. These subclasses may be merged at some future date.

Inherits: Profile The account address. This is either a DNS service
AccountAddress: String (Optional) address (e.g. alice@example.com) or a Mesh Name (@alice).

ServiceUdf: String (Optional) The fingerprint of the service profile to which the account is currently bound.

EscrowEncryption: KeyData (Optional) Escrow key associated with the account.

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

AdministratorSignature: KeyData (Optional) Key used to sign connection assertions to the account.

10.2.11. Structure: ProfileUser

Inherits: ProfileAccount

Account assertion. This is signed by the service hosting the account.

AccountAuthentication: KeyData (Optional) Key used to authenticate requests made under this user account.

AccountSignature: KeyData (Optional) Key used to sign data under the account.

10.2.12. Structure: ProfileGroup

Inherits: ProfileAccount
Describes a group. Note that while a group is created by one person who becomes its first administrator, control of the group may pass to other administrators over time.

[No fields]

10.2.13. Structure: ProfileService

Inherits: Profile
Profile of a Mesh Service

ServiceAuthentication: KeyData (Optional) Key used to authenticate service connections.

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

ServiceSignature: KeyData (Optional) Key used to sign data under the account.

10.2.14. Structure: ProfileHost

Inherits: Profile Key used to authenticate service connections.
KeyAuthentication: KeyData (Optional)
KeyEncryption: KeyData (Optional) Key used to pass encrypted data to the device such as a

10.2.15. Connection Assertions

Connection assertions are used to authenticate and authorize interactions between devices and the service currently servicing the account. They SHOULD NOT be visible to external parties.

10.2.16. Structure: ConnectionDevice

Inherits: Connection
Connection assertion used to authenticate service requests made by a device.

AccountAddress: String (Optional)
The account address

DeviceSignature: KeyData (Optional) The signature key for use of the device under the profile

DeviceEncryption: KeyData (Optional) The encryption key for use of the device under the profile

DeviceAuthentication: KeyData (Optional) The authentication key for use of the device under the profile

10.2.17. Structure: ConnectionApplication

Inherits: Connection
Connection assertion stating that a particular device is

[No fields]

10.2.18. Structure: ConnectionGroup

Describes the connection of a member to a group.

Inherits: Connection
[No fields]

10.2.19. Structure: ConnectionService

Inherits: Connection
[No fields]

10.2.20. Structure: ConnectionHost

Inherits: Connection
[No fields]

10.2.21. Activation Assertions

10.2.22. Structure: ActivationDevice

Contains activation data for device specific keys used in the context of a Mesh account.

Inherits: Activation The UDF of the account
AccountUdf: String (Optional)

10.2.23. Structure: ActivationAccount

Inherits: Activation Grant access to profile online signing key
ProfileSignature: KeyData (Optional) used to sign updates to the profile.

AdministratorSignature: KeyData (Optional) Grant access to Profile administration key used to make changes to administrator catalogs.

AccountEncryption: KeyData (Optional) Grant access to ProfileUser account encryption key

AccountAuthentication: KeyData (Optional) Grant access to ProfileUser account authentication key

AccountSignature: KeyData (Optional) Grant access to ProfileUser account signature key

10.2.24. Structure: ActivationApplication

Inherits: Activation
[No fields]

10.3. Data Structures

Classes describing data used in cataloged data.

10.3.1. Structure: Contact

Inherits: Assertion
Base class for contact entries.

Id: String (Optional) The globally unique contact identifier.

Anchors: Anchor [0..Many] Mesh fingerprints associated with the contact.

NetworkAddresses: NetworkAddress [0..Many] Network address entries

Locations: Location [0..Many] The physical locations the contact is associated with.

Roles: Role [0..Many] The roles of the contact

Bookmark: Bookmark [0..Many] The Web sites and other online presences of the contact

Sources: TaggedSource [0..Many] Source(s) from which this contact was constructed.

10.3.2. Structure: Anchor

Trust anchor

Udf: String (Optional) The trust anchor.

Validation: String (Optional) The means of validation.

10.3.3. Structure: TaggedSource

Source from which contact information was obtained.

LocalName: String (Optional) Short name for the contact information.

Validation: String (Optional) The means of validation.

BinarySource: Binary (Optional) The contact data in binary form.

EnvelopedSource: Enveloped (Optional) The contact data in enveloped form. If present, the BinarySource property is ignored.

10.3.4. Structure: ContactGroup

Inherits: Contact
Contact for a group, including encryption groups.

[No fields]

10.3.5. Structure: ContactPerson

Inherits: Contact List of person names in order of preference
CommonNames: PersonName [0..Many]

10.3.6. Structure: ContactOrganization

Inherits: Contact List of person names in order of preference
CommonNames: OrganizationName [0..Many]

10.3.7. Structure: OrganizationName

The name of an organization

Inactive: Boolean (Optional) If true, the name is not in current use.

RegisteredName: String (Optional) The registered name.

DBA: String (Optional) Names that the organization uses including trading names and doing business as names.

10.3.8. Structure: PersonName

The name of a natural person

Inactive: Boolean (Optional) If true, the name is not in current use.

FullName: String (Optional) The preferred presentation of the full name.

Prefix: String (Optional) Honorific or title, E.g. Sir, Lord, Dr., Mr.

First: String (Optional) First name.

Middle: String [0..Many] Middle names or initials.

Last: String (Optional) Last name.

Suffix: String (Optional) Nominal suffix, e.g. Jr., III, etc.

PostNominal: String (Optional) Post nominal letters (if used).

10.3.9. Structure: NetworkAddress

Provides all means of contacting the individual according to a particular network address

Inactive: Boolean (Optional) If true, the name is not in current use.

Address: String (Optional) The network address, e.g. alice@example.com

NetworkCapability: String [0..Many] The capabilities bound to this address.

EnvelopedProfileAccount: Enveloped (Optional) The account profile

Protocols: NetworkProtocol [0..Many] Public keys associated with the network address

10.3.10. Structure: NetworkProtocol

Protocol: String (Optional) The IANA protocol|identifier of the network protocols by which the contact may be reached using the specified Address.

10.3.11. Structure: Role

OrganizationName: String (Optional)

The organization at which the role is held

Titles: String [0..Many] The titles held with respect to that organization.

Locations: Location [0..Many] Postal or physical addresses associated with the role.

10.3.12. Structure: Location

Appartment: String (Optional)
Street: String (Optional)
District: String (Optional)
Locality: String (Optional)
County: String (Optional)
Postcode: String (Optional)
Country: String (Optional)

10.3.13. Structure: Bookmark

Uri: String (Optional)
Title: String (Optional)
Role: String [0..Many]

10.3.14. Structure: Reference

MessageId: String (Optional) The received message to which this is a response

ResponseId: String (Optional) Message that was generated in response to the original (optional).

Relationship: String (Optional) The relationship type. This can be Read, Unread, Accept, Reject.

10.3.15. Structure: Task

Key: String (Optional) Unique key.

Start: DateTime (Optional)
Finish: DateTime (Optional)
StartTravel: String (Optional)
FinishTravel: String (Optional)
TimeZone: String (Optional)
Title: String (Optional)
Description: String (Optional)
Location: String (Optional)
Trigger: String [0..Many]
Conference: String [0..Many]
Repeat: String (Optional)
Busy: Boolean (Optional)

Labels: String [0..Many] The set of labels describing the entry

10.4. Catalog Entries

10.4.1. Structure: CatalogedEntry

Base class for cataloged Mesh data.

10.4.2. Structure: CatalogedDevice

Inherits: CatalogedEntry
Public device entry, indexed under the device ID Hello

Udf: String (Optional) UDF of the signature key of the device in the Mesh

DeviceUdf: String (Optional) UDF of the offline signature key of the device

SignatureUdf: String (Optional) UDF of the account online signature key

EnvelopedProfileUser: Enveloped (Optional) The Mesh profile

EnvelopedProfileDevice: Enveloped (Optional) The device profile

EnvelopedConnectionUser: Enveloped (Optional) The public assertion demonstrating connection of the Device to the Mesh

EnvelopedActivationDevice: Enveloped (Optional) The activation of the device within the Mesh account

EnvelopedActivationAccount: Enveloped (Optional) The activation of the device within the Mesh account

EnvelopedActivationApplication: Enveloped [0..Many] Application activations granted to the device.

10.4.3. Structure: CatalogedPublication

Inherits: CatalogedEntry
A publication.

Id: String (Optional) Unique identifier code

Authenticator: String (Optional) The witness key value to use to request access to the record.

EnvelopedData: DareEnvelope (Optional) Dare Envelope containing the entry data. The data type is specified by the envelope metadata.

NotOnOrAfter: DateTime (Optional) Epiration time (inclusive)

10.4.4. Structure: CatalogedCredential

Inherits: CatalogedEntry
Protocol: String (Optional)
Service: String (Optional)

Username: String (Optional)
Password: String (Optional)

10.4.5. Structure: CatalogedNetwork

Inherits: CatalogedEntry
Protocol: String (Optional)
Service: String (Optional)
Username: String (Optional)
Password: String (Optional)

10.4.6. Structure: CatalogedContact

Inherits: CatalogedEntry
Key: String (Optional) Unique key.
Self: Boolean (Optional) If true, this catalog entry is for the user who created the catalog.

10.4.7. Structure: CatalogedAccess

Inherits: CatalogedEntry

[No fields]

10.4.8. Structure: CryptographicCapability

Id: String (Optional) The identifier of the capability. If this is a user capability, MUST match the KeyData identifier. If this is a serviced capability, MUST match the value of ServiceId on the corresponding service capability.

KeyData: KeyData (Optional) The key that enables the capability

EnvelopedKeyShares: Enveloped [0..Many] One or more enveloped key shares.

SubjectId: String (Optional) The identifier of the resource that is controlled using the key.

SubjectAddress: String (Optional) The address of the resource that is controlled using the key.

10.4.9. Structure: CapabilityDecrypt

Inherits: CryptographicCapability

The corresponding key is a decryption key

[No fields]

10.4.10. Structure: CapabilityDecryptPartial

Inherits: CapabilityDecrypt

The corresponding key is an encryption key

ServiceId: String (Optional)
The identifier used to claim the capability from the service.[Only present for a partial capability.]

ServiceAddress: String (Optional) The service account that supports a serviced capability. [Only present for a partial capability.]

10.4.11. Structure: CapabilityDecryptServiced

Inherits: CapabilityDecrypt
The corresponding key is an encryption key

AuthenticationId: String (Optional) UDF of trust root under which request to use a serviced capability must be authorized. [Only present for a serviced capability]

10.4.12. Structure: CapabilitySign

Inherits: CryptographicCapability
The corresponding key is an administration key

[No fields]

10.4.13. Structure: CapabilityKeyGenerate

Inherits: CryptographicCapability
The corresponding key is a key that may be used to generate key shares.

[No fields]

10.4.14. Structure: CapabilityFairExchange

Inherits: CryptographicCapability
The corresponding key is a decryption key to be used in accordance with the Micali Fair Electronic Exchange with Invisible Trusted Parties protocol.

[No fields]

10.4.15. Structure: CatalogedBookmark

Inherits: CatalogedEntry
Uri: String (Optional)
Title: String (Optional)
Path: String (Optional)

10.4.16. Structure: CatalogedTask

Inherits: CatalogedEntry

EnvelopedTask: Enveloped (Optional)
Title: String (Optional) Unique key.
Key: String (Optional)

10.4.17. Structure: CatalogedApplication

Inherits: CatalogedEntry Enveloped keys for use with Application
Key: String (Optional)
EnvelopedCapabilities: DareEnvelope [0..Many]

10.4.18. Structure: CatalogedMember

ContactAddress: String (Optional)
MemberCapabilityId: String (Optional)
ServiceCapabilityId: String (Optional)
Inherits: CatalogedEntry

10.4.19. Structure: CatalogedGroup

Inherits: CatalogedApplication The Mesh profile
EnvelopedProfileGroup: Enveloped (Optional)
EnvelopedActivationAccount: Enveloped (Optional) The activation of the device within the Mesh account

10.4.20. Structure: CatalogedApplicationSSH

Inherits: CatalogedApplication
[No fields]

10.4.21. Structure: CatalogedApplicationMail

Inherits: CatalogedApplication
[No fields]

10.4.22. Structure: CatalogedApplicationNetwork

Inherits: CatalogedApplication
[No fields]

10.5. Publications

10.5.1. Structure: DevicePreconfiguration

A data structure that is passed

EnvelopedProfileDevice: Enveloped (Optional) The device profile

EnvelopedConnectionDevice: Enveloped (Optional) The device connection

ConnectUri: String (Optional) The connection URI. This would normally be printed on the device as a QR code.

10.6. Messages

10.6.1. Structure: Message

MessageId: String (Optional) Unique per-message ID. When encapsulating a Mesh Message in a DARE envelope, the envelope EnvelopeID field MUST be a UDF fingerprint of the MessageId value.

Sender: String (Optional)
Recipient: String (Optional)

10.6.2. Structure: MessageError

Inherits: Message
ErrorCode: String (Optional)

10.6.3. Structure: MessageComplete

Inherits: Message
References: Reference [0..Many]

10.6.4. Structure: MessagePinValidated

Inherits: Message Enveloped data that is authenticated by means of
AuthenticatedData: DareEnvelope (Optional) the PIN

ClientNonce: Binary (Optional) Nonce provided by the client to validate the PIN

PinId: String (Optional) Pin identifier value calculated from the PIN code, action and account address.

PinWitness: Binary (Optional) Witness value calculated as KDF (Device.Udf + AccountAddress, ClientNonce)

10.6.5. Structure: MessagePin

Account: String (Optional) If true, authentication against the PIN
Inherits: Message code is sufficient to complete the associated
Expires: DateTime (Optional) action without further authorization.
Automatic: Boolean (Optional)
SaltedPin: String (Optional) PIN code bound to the specified action.

Action: String (Optional) The action to which this PIN code is bound.

10.6.6. Structure: RequestConnection

Connection request message. This message contains the information

Inherits: MessagePinValidated
AccountAddress: String (Optional)

10.6.7. Structure: AcknowledgeConnection

Connection request message generated by a service on receipt of a valid MessageConnectionRequestClient

Inherits: Message The client connection request.
EnvelopedRequestConnection: Enveloped (Optional)
ServerNonce: Binary (Optional)
Witness: String (Optional)

10.6.8. Structure: RespondConnection

Respond to RequestConnection message to grant or refuse the connection request.

Inherits: Message The response to the request. One of "Accept",
Result: String (Optional) "Reject" or "Pending".

CatalogedDevice: CatalogedDevice (Optional) The device information. MUST be present if the value of Result is "Accept". MUST be absent or null otherwise.

10.6.9. Structure: MessageContact

Inherits: MessagePinValidated If true, requests that the recipient
Reply: Boolean (Optional) return their own contact information in reply.

Subject: String (Optional) Optional explanation of the reason for the request.

PIN: String (Optional) One time authentication code supplied to a recipient to allow authentication of the response.

10.6.10. Structure: GroupInvitation

Inherits: Message
Text: String (Optional)

10.6.11. Structure: RequestConfirmation

Inherits: Message
Text: String (Optional)

10.6.12. Structure: ResponseConfirmation

Inherits: Message
Request: Enveloped (Optional)
Accept: Boolean (Optional)

10.6.13. Structure: RequestTask

Inherits: Message

[No fields]

10.6.14. Structure: MessageClaim

Inherits: Message
PublicationId: String (Optional)

ServiceAuthenticate: String (Optional)
DeviceAuthenticate: String (Optional)
Expires: DateTime (Optional)

10.6.15. Structure: ProcessResult

For future use, allows logging of operations and results

Inherits: Message The error report code.
Success: Boolean (Optional)
ErrorReport: String (Optional)

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. Appendix A: Example Container Organization (not normative)

The means by which profiles are stored on devices is outside the scope of the specification. Only catalogs that are required to be shared between machines by means of accounts need to be standardized.

14.1. Device

Host Catalog: Host.dare Catalog of all the Mesh Profiles that the user has registered with the device and the latest version of the device profile for this device.

MeshCatalog: [UDF-Mesh].dcat Catalog containing the Account Entries for the Mesh [UDF-Mesh].

Account Catalogs: [UDF-Account]/mmm_Device.dcat The device catalog associated with the specified account

Account Catalogs: [UDF-Account]/[Catalog name].dcat The set of account catalogs that are shared verbatim between the devices connected to the account.

14.1.1. Creating a new Mesh

Create new Mesh Profile, Device Profile, Add to Host Catalog

Create MeshCatalog

14.1.2. Adding an Account

Create new Account Profile, Add to MeshCatalog

Create new Account Device Catalog

For each device to be added to the account: Create Account Connection Assertion, add to Account Device Catalog.

14.1.3. Adding a Device

Create a Device Connection Assertion.

For each account the device is to be added to: Create Account Connection Assertion, add to Account Device Catalog.

14.2. Service

Master Catalog Catalog of all services on machine

Service Catalog Catalog of accounts in the service.

14.2.1. Creating a Service

Create a Service Description, add to Master Catalog

14.2.2. Adding an Account

Create the account entry, add to Service Catalog

Create the Account Directory

15. Appendix B: Collected Authentication and Encryption Requirements

15.1. Mesh Messaging

Message	Signer	Recipients
RequestConnection	Device	Service
AcknowledgeConnection	Service	Device, Receiver
OfferGroup	Sender	Receiver
RequestContact	Sender	Receiver
ReplyContact	Sender	Receiver
RequestConfirmation	Sender	Receiver
ResponseConfirmation	Sender	Receiver

Message	Signer	Recipients
RequestTask	Sender	Receiver
ResponseTask	Sender	Receiver

Table 1

16. 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-16, 13 January 2021, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-architecture-16>.

[draft-hallambaker-mesh-cryptography]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part VIII: Cryptographic Algorithms", Work in Progress, Internet-Draft, draft-hallambaker-mesh-cryptography-07, 2 November 2020, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-cryptography-07>.

[draft-hallambaker-mesh-dare]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part III : Data At Rest Encryption (DARE)", Work in Progress, Internet-Draft, draft-hallambaker-mesh-dare-11, 13 January 2021, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-dare-11>.

[draft-hallambaker-mesh-discovery]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part VI: Mesh Discovery Service", Work in Progress, Internet-Draft, draft-hallambaker-mesh-discovery-01, 13 January 2021, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-discovery-01>.

[draft-hallambaker-mesh-notary] "[Reference Not Found!]".

[draft-hallambaker-mesh-protocol]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part V: Protocol Reference", Work in Progress, Internet-Draft, draft-hallambaker-mesh-protocol-08, 13 January 2021, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-protocol-08>.

[draft-hallambaker-mesh-security]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part VII: Security Considerations", Work in Progress, Internet-Draft, draft-hallambaker-mesh-security-06, 2 November

2020, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-security-06>.

[draft-hallambaker-mesh-udf]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part II: Uniform Data Fingerprint.", Work in Progress, Internet-Draft, draft-hallambaker-mesh-udf-12, 13 January 2021, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-udf-12>.

[draft-hallambaker-threshold]
Hallam-Baker, P., "Threshold Modes in Elliptic Curves", Work in Progress, Internet-Draft, draft-hallambaker-threshold-05, 13 January 2021, <https://datatracker.ietf.org/doc/html/draft-hallambaker-threshold-05>.

[draft-hallambaker-threshold-sigs]
Hallam-Baker, P., "Threshold Signatures in Elliptic Curves", Work in Progress, Internet-Draft, draft-hallambaker-threshold-sigs-06, 13 January 2021, <https://datatracker.ietf.org/doc/html/draft-hallambaker-threshold-sigs-06>.

[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>.

17. Informative References

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

[RFC2426] Dawson, F. and T. Howes, "vCard MIME Directory Profile", RFC 2426, DOI 10.17487/RFC2426, September 1998, <https://www.rfc-editor.org/rfc/rfc2426>.

[RFC5545] Desruisseaux, B., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009, <https://www.rfc-editor.org/rfc/rfc5545>.