Mathematical Mesh 3.0 Part IV: Schema Reference

Mesh Schema Reference

<series>draft-hallambaker-mesh-schema

<status>informational

<stream>independent

<ipr>trust200902

<author>Phillip Hallam-Baker

<surname>Hallam-Baker

<initials>P. M.

<firstname>Phillip

<email>phill@hallambaker.com

<organization>ThresholdSecrets.com

<also>http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html

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.

# 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* <norm="draft-hallambaker-mesh-architecture"/>. For information on the implementation of the Mesh Service protocol, consult the accompanying *Protocol Reference* <norm="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 <info="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.

# 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.

## 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 <norm="RFC2119"/>.

## Defined Terms

The terms of art used in this document are described in the *Mesh Architecture Guide* <norm="draft-hallambaker-mesh-architecture"/>.

## Related Specifications

The architecture of the Mathematical Mesh is described in the *Mesh Architecture Guide* <norm="draft-hallambaker-mesh-architecture"/>. The Mesh documentation set and related specifications are described in this document.

## Implementation Status

The implementation status of the reference code base is described in the companion document <info="draft-hallambaker-mesh-developer"/>.

# 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.

## 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 keys, AccountSignature and AccountAuthentication which are intended for future use to 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.

<include="..\Examples\SchemaAliceProfile.md">

## 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:

<include="..\Examples\SchemaAliceDeviceCoffee.md">

### 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:

<include="..\Examples\SchemaAliceActivationCoffee.md">

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 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.

<include="..\Examples\SchemaAliceActivationWatch.md">

## Service

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

<include="..\Examples\SchemaProfileService.md">

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.

<include="..\Examples\SchemaConnectionHost.md">

# 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 <norm="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.

## 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.

The access catalog is unique in that it is the only catalog that can be read by the Mesh Service which is the access control enforcement point for the actions specified in the catalog.

[Should we call the catalog mmm\_Service? How about mmm\_Service@example.com??]

## 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.

### 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.

### 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.

<include=..\Examples\SchemaEntrySSH.md>

### 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.

<include=..\Examples\SchemaEntryMail.md>

## 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.

<include=..\Examples\SchemaEntryBookmark.md>

## Contact

The contact catalog mmm\_contact contains CatalogEntryContact entries which describe

<include=..\Examples\SchemaEntryContact.md>

The fields of the contact catalog provide a superset of the capabilities of vCard <info="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.

## Credential

The credential catalog mmm\_credential contains CatalogEntryCredential entries which describe credentials used to access network resources.

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.

<include=..\Examples\SchemaEntryCredential.md>

## Device

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

## Network

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

<include=..\Examples\SchemaEntryNetwork.md>

## Publication

The publication catalog mmm\_Publication contains CatalogEntryPublication entries which describe content published through the account.

## 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 <info="RFC5545"/>. Specification of additional fields to allow task triggering on geographic location and/or completion of other tasks is a work in progress.

<include=..\Examples\SchemaEntryTask.md>

# 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.

## 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.

## 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.

## 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 state Initial, Closed:

Initial

The initial state of a message posted to the spool.

Closed

The action associated with the message has been completed.

# 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.

## Key Derivation from Seed

Mesh Keys that derived from a seed value use the mechanism described in <norm="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:

<include="..\Examples\SchemaDeriveTables.md">

## Message Response Identifier

Every Mesh message has a unique MessageId. When encapsulated in a DARE Envelope, the EnvelopeId field of the envelope header is the UDF Content Digest of the enclosed MessageId as a string:

public static string GetEnvelopeId(string messageID) =>

UDF.ContentDigestOfUDF(messageID);

When processing a Mesh message results in the creation of a response to the sender, the MessageId of the response is UDF Content Digest of the Binary Data Sequence of the original MessageId:

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),

};

<include="..\Examples\SchemaMessageIds.md">

## 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:

|  |  |  |
| --- | --- | --- |
| **Code** | **Mesh Message** | **Purpose** |
| "MessageContact" | MessageContact | Contact exchange |
| "RequestConnection" | RequestConnection | Device connection |

These inputs are used to derive values as follows:

alg = UdfAlg (PIN)

pinData = UdfBDS (PIN)

saltedPINData = MAC (Action, pinData)

saltedPIN = UDFPresent (Authenticator\_HMAC\_SHA\_2\_512 + saltedPINData)

PinId = UDFPresent (MAC (Account, saltedPINData))

witnessData = Account.ToUTF8() + ClientNonce + PayloadDigest

witnessValue = MAC (witnessData , saltedPINData)

<include="..\Examples\SchemaPINFunction.md">

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.

<include="..\Examples\SchemaPINWitness.md">

## 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 <norm="draft-hallambaker-mesh-udf"/>.

## 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.

<include="..\Examples\SchemaClientAuthKeyAgreement.md">

## 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 <norm="draft-hallambaker-threshold"/> and <norm="draft-hallambaker-threshold-sigs"/>.

# 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 one profile to another.

<figuresvg="../Images/ProfilesConnections.svg">Profiles And Connections

## 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 <norm="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.

## 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.

## 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.

# Architecture

The Mesh architecture has four principal components:

Mesh Device Management

Binds a collection of devices that the owner of the Mesh uses together to function as a single personal Mesh.

Mesh Account

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

Mesh Service

Provides a service identifier (e.g. alice@example.com) through which devices and other Mesh users may interact with a Mesh Account.

Mesh Messaging

Allows short messages (less than 32KB) to be exchanged between Mesh devices connected to an account and between Mesh Accounts.

Device management and Accounts components are defined by a data schema alone. The Service and Messaging components are defined by a data schema and a service protocol.

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 account service provider which the user may change at any time of their choosing. A Mesh account may be connected to multiple service providers to provide backup capability or to none.

For example, Alice's personal Mesh has one Master Profile, multiple device profiles (two of which are shown here) and two Account Profiles. Her Personal account is connected to two Mesh services while her Business account is not currently connected to any service:

<figuresvg="../Images/ProfileMaster.svg">Alice's Personal Mesh

In normal circumstances, a user will create a personal Mesh and add their first device, account and service at once. These are shown here as separate operations to illustrate the separation of the Mesh components.

## Device Management

Device Management provides the foundation for all Mesh functions allowing a collection of devices belonging to a user to function as a single personal Mesh.

The device management layer of a personal Mesh consists of exactly one Master Profile and a catalog containing the entries describing the connected devices.

### Master Profile

A Mesh master 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.

A Master Profile MAY contain one or more MasterEscrowKeys. These MAY be used to escrow private keys used for encryption. They SHOULD NOT be used to escrow authentication keys and MUST NOT be used to escrow signature keys.

<figuresvg="../Images/ProfileMasterKeys.svg">Master Profile and Associated Device and Account Connection Assertions.

A user should not need to replace their master profile unless they intend to establish a separate identity. To minimize the risk of disclosure, the Master Signature Key is only ever used to sign updates to the master profile itself. This allows the user to secure their Master 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.

Alice creates a ProfileMaster with one administration key and one master escrow key:

<include=..\Examples\SchemaMaster.md>

#### Creating a ProfileMaster

Creating a ProfileMaster comprises the steps of:

1. Creating a Master Signature key.
2. Creating an Online Signing Key
3. Signing the ProfileMaster using the Master Signature Key
4. Persisting the ProfileMaster on the administration device to the CatalogHost.
5. (Optional) Connecting at least one Administration Device and granting it the ActivationAdministration activation.

#### Updating a ProfileMaster

Updating a ProfileMaster comprises the steps of:

1. Making the necessary changes.
2. Signing the ProfileMaster using the Master Signature Key
3. Persisting the ProfileMaster on the administration device to the CatalogHost.

#### The Device Catalog

Each personal Mesh 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.

### 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:

<figuresvg="../Images/ProfileDeviceKeys.svg">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.

<include=..\Examples\SchemaDevice.md>

The derivation of the Connection encryption and signature keys from the Profile and Private contributions in this example is shown in <norm="draft-hallambaker-mesh-cryptography"/>.

#### Creating a ProfileDevice

Creating a ProfileDevice comprises the steps of:

1. Creating the necessary key
2. Signing the ProfileDevice using the Master Signature Key

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

#### Connection to a Personal Mesh

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

1. Generating the PrivateDevice keys.
2. Creating the ConnectionDevice data from the public components of the ProfileDevice and PrivateDevice keys and signing it using the administration key.
3. Creating the Activations for the device and signing them using the administration key.
4. Creating the CatalogEntryDevice for the device and adding it to the CatalogDevice of the Personal Mesh.
5. If the Personal Mesh has accounts that are connected to a Mesh Service, synchronizing the CatalogEntryDevice to those services.

## Mesh Accounts

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

A Mesh Profile MAY be connected to multiple accounts at the same time allowing the user to maintain separate personas for separate purposes.

Unlike traditional Internet application accounts, Mesh accounts are created by and belong to the user, not the Mesh Service provider. A user MAY change their Mesh Service provider at any time and disconnect the profile from all Mesh Services (e.g. to archive the account).

Alice's personal account is connected to two Mesh services:

<figuresvg="../Images/ProfileAccount.svg">Account Profile Connected to Devices and Services.

<include=..\Examples\SchemaAccount.md>

### Creating a ProfileAccount

Creating a ProfileAccount comprises the steps of:

1. [TBS]
2. .
3. Signing the ProfileMaster using the Master Signature Key

### Connecting a Device to an Account

Adding a device to an account comprises the steps of:

1. Creating a PrivateAccount instance for the device.
2. Creating a ConnectionAccountDevice for the device using the public keys from the PrivateAccount instance and the ProfileDevice.
3. Creating an ActivationAccount for the device containing the PrivateAccount and ConnectionAccountDevice instances.
4. Adding the ActivationAccount to the CatalogEntryDevice of the device.
5. If the Personal Mesh has accounts that are connected to a Mesh Service, synchronizing the CatalogEntryDevice to those services.

### Binding and Account to a Service

Binding a ProfileAccount to a Mesh Service the steps of:

1. [TBS]
2. .
3. Signing the ProfileMaster using the Master Signature Key

## Mesh Services

A service profile provides the axiom of trust and cryptographic keys for a Mesh Service. A Mesh Service Host SHOULD return a copy of its ProfileHost and the parent ProfileService in response to a Hello transaction request.

<figuresvg="../Images/ProfileService.svg">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.

<include=..\Examples\SchemaService.md>

### Creating a ProfileService

[TBS]

Creating a ProfileService comprises the steps of:

1. [TBS]
2. .
3. [TBS]
4. Signing the ProfileMaster using the Master Signature Key

### Creating a ProfileHost

Creating a ProfileHost comprises the steps of:

1. [TBS]
2. .
3. [TBS]
4. Signing the ConnectionHost using the Master Signature Key of the ProfileService.

### Creating a ConnectionHost

Creating a ConnectionHost comprises the steps of:

1. [TBS]
2. .
3. Signing the ConnectionHost using the Master Signature Key of the ProfileService.

## 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).

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

<figuresvg="../Images/FourCorner.svg">Performing Access Control on Outbound Messages

The outbound Mesh Service checks to see that the message request 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.

<figuresvg="../Images/OutboundAccessControl.svg">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.

<figuresvg="../Images/InboundAccessControl.svg">Performing Access Control on Inbound Messages

### 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 naïve 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.

# Mesh Messages

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 <norm="draft-hallambaker-mesh-protocol"/>.

## Completion

Completion messages are dummy messages that are added to a Mesh Spool to change the status of messages previously posted. Any message that is in the inbound spool and has not been erased or redacted MAY be marked as read, unread or deleted. Any message in the outbound spool MAY be marked as sent, received or deleted.

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 Service will purge the message however.

<include=..\Examples\SchemaMessageCompletion.md>

## Connection

Connection requests are sent by a device requesting connection to a Mesh Service Account.

The MessageConnectionRequest is originally sent by the device requesting connection to the Mesh Service associated with the account.

If the connection request is accepted by the Mesh Service, it creates a MessageConnectionResponse containing the ServerNonce and Witness values used in the authentication of the response together with a verbatim copy of the original request. The MessageConnectionResponse is then returned to the device that made the original request and placed on the SpoolInbound of the account to which the request was directed.

Further details of this mechanism are described in <norm="draft-hallambaker-mesh-protocol"/>.

<include=..\Examples\SchemaMessageConnection.md>

## Contact

A contact request presents a proposed contact entry and requests that it be added to the Contacts catalog of the specified Mesh Service Account. A contact request is usually sent by the party requesting that their contact be added but this is not necessarily the case.

The MessageContact contains a DARE Envelope containing the Contact information of the requester. If the request is accepted, this information will be added to the contact catalog of the relevant account. If the Reply field has the value 'true', this indicates that the sender is asking for the recipient to return their own credentials in reply.

Since the sender requires the user's contact information before the request can be made, the MessageContact message MAY be encrypted under either the user's account encryption key (if known) or the Mesh Service encryption key (which may be obtained from the service on request.

<include=..\Examples\SchemaMessageContact.md>

The current protocol assumes that all contact management will be performed end-to-end through the Mesh Services themselves. If the number of Mesh users were to become very large, additional infrastructure to facilitate contact management will be required. These topics are discussed at a high level in <info="draft-hallambaker-mesh-trust"/>.

In situations where a user is well known and has a very large number of contacts, they are likely to make use of a tiered approach to contact management in which they keep separate accounts for their 'public' and 'restricted' personas and delegate management of their public account to a subordinate or to their Mesh Service provider.

## Confirmation

Confirmation messages are used to provide an improved form of second factor authentication capability.

Two confirmation messages are specified, a request and response.

A confirmation request is initiated by sending a MessageConfirmationRequest to the Mesh Service hosting the recipient Mesh Service Account. The request specifies the question that is to be put to the user.

To respond to a confirmation request, a user generates a MessageConfirmationResponse. This MUST be signed by a device authorized to respond to confirmation requests by a Device Connection Assertion with the Confirmation privilege.

<include=..\Examples\SchemaMessageConfirmation.md>

# Schema

<include=..\Generated\SchemaMesh.md>

# Security Considerations

The security considerations for use and implementation of Mesh services and applications are described in the Mesh Security Considerations guide <norm="draft-hallambaker-mesh-security"/>.

# IANA Considerations

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

# Acknowledgements

A list of people who have contributed to the design of the Mesh is presented in <norm="draft-hallambaker-mesh-architecture"/>.

# 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.

## 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.

### Creating a new Mesh

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

### 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.

### 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.

## Service

Master Catalog

Catalog of all services on machine

Service Catalog

Catalog of accounts in the service.

### Creating a Service

* Create a Service Description, add to Master Catalog

### Adding an Account

* Create the account entry, add to Service Catalog
* Create the Account Directory

# Appendix B: Collected Authentication and Encryption Requirements

## 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 |
| RequestTask | Sender | Receiver |
| ResponseTask | Sender | Receiver |