Mathematical Mesh 3.0 Part V: Protocol Reference

Mesh Protocol Reference

<series>draft-hallambaker-mesh-protocol

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

Mesh Service Accounts support the following services:

* Provides the master persistence store for the Catalogs and Spools associated with the account.
* Enables synchronization of Catalogs and Spools with connected devices.
* Enforces access control on inbound Mesh Messages from other users and other Mesh Services.
* Authenticates outbound Mesh Messages, certifying that they comply with abuse mitigation policies.

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

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

Hello

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

CreateAccount, DeleteAccount

Manage the creation and deletion of accounts at the service.

Status, Download, Upload

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

Connect

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

Post

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

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

# 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"/>.

# Mesh Protocols

## Mesh Service Protocol

## Mesh Messaging Protocols

# Mesh Service

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

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

Data Loss

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

Integrity (Stale Data)

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

Messaging

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

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

Traffic analysis

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

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

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

## Data Model

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

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

Account Catalog

Contains the account entries.

Incident Spool

Reports of potential abuse

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

## Partitioning

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

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

# Protocol Bindings

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

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

## DNS Web Service Discovery

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

* Service Name: mmm
* DNS Prefix: \_mmm.\_tcp
* Well Known service suffix: /.well-known/mmm

## Web Service Protocol Binding

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

* Discovery: DNS Service discovery
* Transport: TLS
* Application: HTTP
* Presentation: DARE Message
* Encoding: JSON, JSON-B

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

### Transport Security

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

TLS version 1.3 <norm="RFC8446"/> or higher MUST be supported. Client authentication SHOULD NOT be used.

### HTTP Message Binding

All messages are exchanged as HTTP POST transactions. Support for and use of HTTP/1.1 <norm="RFC7230"/> is REQUIRED. Services MAY support HTTP/2.

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

### Request

The HTTP request MAY contain any valid HTTP header specified in <norm="RFC7230"/>.

Request Line URI

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

Request Line Method

POST

Host: Header

&<domain>

Content-Encoding

As specified in section yy below.

Content-Type

As specified in section zz below.

Content-Length or Transfer-Encoding

As specified in <norm="RFC7230"/>.

Payload

The content payload as specified in section XX below.

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

### Response

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

Response Code

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

Content-Type

As specified in section zz below.

Content-Length or Transfer-Encoding

As specified in <norm="RFC7230"/>.

Cache-Control

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

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

## DARE Message Encapsulation

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

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

Mesh Service MUST support the use of Mutually Authenticated Key Exchange <norm="draft-hallambaker-mesh-security"/> to establish the Master Key used for authentication of requests and responses.

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

### Null Authentication

Null Authentication MAY be used to make a Hello Request.

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

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

### Device Authentication

Device Authentication is used in two circumstances

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

### Profile Authentication

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

### Ticket Authentication

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

## Payload Encoding

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

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

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

* The ability to use arbitrary strings as field names.
* The use of JSON objects to define maps directly

The following data field types are used:

Integer

Integer values are encoded as JSON number values.

String

Test strings are encoded as JSON text strings.

Boolean

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

Sequence

Sequences of data items that are encoded as JSON arrays

Object of known type

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

Object of variable type

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

Binary Data

Byte sequences are converted to BASE64-url encoding <norm="RFC4648"/> and encoded as JSON string values.

Date Time

Date Time values are converted to Internet time format as described in <norm="RFC3339"/> and encoded as JSON string values.

## Error handling and response codes

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

* Service Layer
* HTTP Layer
* Transport Layer

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

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

# Mesh Service Transactions

## Service Description

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

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

## Account Creation

### Bind User Account

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

The BindRequest message specifies the account address and ProfileUser of the account to be serviced.

The BindAccount transaction is unique in that it can fail to complete for reasons that are outside the scope of the Mesh specifications. Creation of an account might require payment to be made or authentication of the user's credentials. It is thus quite normal for the result of a CreateRequest to be the account being created in an 'on hold' state which can only be changed out of band.

If the request is at least partially successful, a BindResponse message is returned. In the case of partial success, a description of the request status and link to a Web page providing further details MAY be returned.

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

[Future: Consider converting this to a Messaging flow.]

### Bind Group Account

Mesh Group Accounts are created in the same manner as user accounts except that a ProfileGroup is specified.

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

### Unbind Account

An account registration is deleted using theUnbindAccount transaction.

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

## Persistence Store Management

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

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

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

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

### Status

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

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

### Download

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

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

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

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

### Conflict Detection

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

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

### Filtering

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

### Transact

The transact transaction appends envelopes to one or more stores. The operation is atomic, that is either all the changes specified will be made to the stores or none will. This ensures that simultaneous attempts to update a store do not result in race conditions.

Each update to a catalog or container specifies the expected container index and apex digest. This provides a strong guarantee of consistency. The service MUST verify each update to check that the Merkle Tree values specified are consistent with the store entries and that the signature on the apex value (if specified) is valid and correct.

Services MAY impose limits on the size and number of additions performed in response to a TransactRequest message to ensure that processing time does not degrade performance for other users.

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

## Messaging

Mesh Messaging is an asynchronous messaging service that allows exchange of information between devices connected to a Mesh account and between Mesh users.

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

<figuresvg="../Images/ArchFourCorner2.svg">

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

Client-Service (Post Transaction)

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

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

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

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

Service-Service (Post Transaction)

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

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

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

*Denial of Service Mitigation*

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

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

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

*Access Control*

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

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

The accreditation of the sender

The accreditation of the transmitting Service

The type of Mesh Message being sent

User policy as specified in their Contact Catalog

Site policy.

Service-Client (Synchronization)

The final recipient receives the message by synchronizing their device. The message received will be appended to the inbound spool.

## Publication

The Publication mechanism allows content to be published through a Mesh Account and retrieved by means of the EARL mechanism described in Uniform Data Fingerprint <norm="draft-hallambaker-mesh-udf"/>. This mechanism is used in certain flows supported by the Mesh Device Connection and Contact Exchange functions.

Content is published by appending an entry to an account's Publication spool. The content may then be retrieved by issuing a claim to the account specifying the publication identifier that is authenticated under the value specified in the EARL.

Use of the Publication spool to post content necessarily requires that the content be smaller than the maximum message size imposed by the Mesh Service so that it can be uploaded to the service by means of a Transact transaction.

Publication of large data items will require modification of the protocol to support use of a detached message body. Transfer of a detached message body is outside the scope of this document.

### Claim

The Claim Transaction is used to obtain the publication from the service. The claim request contains a MessageClaim signed by the party requesting the device. This in turn contains a proof of knowledge of the authentication PIN that can be verified by the content creator and a proof of knowledge of the authentication PIN that can be verified by the service.

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

### Poll Claim

The static device connection protocol allows a device connected to an account to retrieve the latest claim made for a particular publication. This is used in the device connection protocol.

The device polling the service specifies the identifier of the publication it is attempting to obtain the claim for.

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

## Cryptographic

The Operate transaction is used to perform one or more cryptographic operations using private key material recorded in the Threshold Catalog. Such operations typically represent one part of a threshold key operation divided between the service and a device connected to an account.

As with all operations involving the Threshold catalog, the request MUST meet the authentication criteria specified by the catalog entry. These typically include the request being authenticated by a specific key.

### Generate Key Shares

CryptographicOperationShare is used to request that a private key held by the service to be divided into two or more key shares. One key share is then encrypted under the encryption key of the service and the others are encrypted under public keys specified in the request. These parameters are returned in a CryptographicResultShare.

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

### Key Agreement

CryptographicOperationKeyAgreement is used to request a threshold key agreement operation on a specified public key.

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

### Sign

Threshold signature is not currently supported.

# Message Transactions

Message transactions are interaction between devices connected to the same account and between accounts.

All messages are signed by the sender and encrypted under the encryption key of the recipient if this is known to the sender.

## PIN Code

The PIN Code Message Transaction is used to register and validate PIN codes used to authenticate other message transactions. This is currently used as an option in the Device Connection and Contact Exchange transactions.

Derivation of the SaltedPin, MessageId and Witness values from their respective inputs is described in the Schema Reference <norm="draft-hallambaker-mesh-schema"/>.

### Registration

To register a PIN code to an Account, a device:

* Generates the PIN code value
* Calculates the SaltedPin value for the specified Action
* Calculates the PinId binding the specified SaltedPin to the Account.
* Creates and signs MessagePin containing the SaltedPin , Action and Account values with the MessageId value PinId.
* Appends the MessagePin value to the Administration Spool of the Account.

Note that this construction provides limited protection against forgery attacks by a party with access to the MessagePin. A party with such access can use it to construct the witness value required to authenticate a request.

PIN Code values consist of an opaque sequence of octets represented as a UDF nonce value. Codes are presented in canonical UDF form, i.e. Base32 encoding separated into groups of 4 characters. The PIN value is converted to binary form for calculation of the SaltedPin, thus ensuring that the canonical form of the PIN value is used.

### Authentication

The PIN Code value is passed out of band to a user who will enter it into a device to authenticate a request made to the issuer.

A request that MAY be validated by means of a PIN is a subclass of MessagePinValidated and contains the following fields:

AuthenticatedData

A DARE Envelope containing the data that is authenticated.

ClientNonce

A nonce value used to prevent certain replay attacks.

PinId

Digest value binding the SaltedPin to the Account.

PinWitness

Witness value calculated as KDF (Device.UDF + AccountAddress, ClientNonce)

The device uses the PIN code and Action identifier corresponding to the desired request to calculate the SaltedPin value in the same manner as during registration. This value is then used to calculate the PinId and PinWitness values.

### Validation

The PIN code is validated by performing the steps of:

* Calculating the SaltedPin value from the PIN code and Action
* Calculating PinId from SaltedPin and Account
* Retrieving a MessagePin from the Administration spool with the MessageId PinId.
* Calculating the PinWitness value from SaltedPin, ClientNonce and AuthenticatedData and checking this matches the value specified in the message.
* Performing the requested action.
* Posting a Complete message to the Administration Spool of the Account marking the PIN code as used.

This process can fail at multiple points resulting in different error results:

PinInvalid

No PIN code is specified, the Pin code indicates an unsupported algorithm or the calculated PinWitness does not match the one specified by the request.

PinUsed

The PIN code has been used previously.

PinExpired

The PIN code is no longer valid.

Note that in the case that an attempt is made to reuse a PIN, it is not automatically the case that the first use of the PIN was the one that was valid and only the second attempt was invalid. Implementations SHOULD alert the user to the attempted re-use so that this possibility can be considered and appropriate action taken.

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

## Contact Exchange

The contact exchange transaction is used to support unilateral or mutual exchange of contact information. Contact exchange has three functions in the Mesh:

* To exchange public key information to allow encryption of messages sent to and verification of signatures on messages sent from the contact subject.
* To exchange contact information allowing use of other communication protocols (e.g. telephone, SMS, xmpp, SMTP, OpenPGP, S/MIME, etc).
* To request that the recipient grant privileges to accept certain types of messages from the contact subject.

Registration of the subject's contact information in the Mesh Naming Service eliminates the need for the first of these functions but not the other two. To prevent abuse, every Mesh Message is subject to access control and a Mesh service will only accept a message from a sender if there is an entry in the Threshold Catalog of the account that expressly permits delivery of messages of the specified type that are authenticated by an authorized signature key.

### Remote

The Remote Contact Exchange transaction consists of a sequence of MessageContact messages sent from the initiator to the responder, responder to the initiator, etc. While there is in principle no limit on the number of messages exchanged, most exchanges will be completed in three exchanges or less:

Initiator to Responder

Contains Initiator contact data without authentication context from the exchange.

Responder to Initiator (optional)

Contains Responder contact data authenticated under a PIN challenge presented in the previous message.

Initiator to Responder (optional)

Contains Initiator contact data authenticated under a PIN challenge presented in the previous message.

Each message provides the recipient with additional information which MAY motivate the recipient to provide additional contact information to the sender.

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

The Mesh Contact Exchange transaction does not provide for validation of the contact information beyond the binding to the Mesh Account Address used to perform the exchange.

### PIN

Exchange of a PIN code out of band allows the initial MessageContact to be authenticated. This mode of authentication is particularly suited to in-person exchange of credentials where the PIN code and other information required to complete the transaction are passed by some means of short range communication such as Bluetooth or presentation of a QR code. In either case, the connection information is presented in the form of a URI combining the type of interaction (contact exchange), the contact address and the authentication data.

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

### EARL

A MessageContact message MAY be published as an EARL. This allows contact data to be presented to the recipient on a printed document such as a business card in machine readable format such as a QR code.

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

## Group Invitation

The GroupInvitation message is used to invite a recipient to join a Mesh Group. The message specifies the group name and the contact entry for the group. The contact entry includes the CapabilityDecryptServiced used to decrypt messages sent to the group when combined with information provided by the threshold service for the group.

Receipt of a GroupInvitation message does not require a response.

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

## Confirmation

The confirmation transaction consists of a RequestConfirmation message from the initiator followed by a ResponseConfirmation from the responder.

The RequestConfirmation message specifies the action that is requested.

The ResponseConfirmation message contains the enveloped RequestConfirmation message signed by the initiator and the disposition of the responder, Accept = true if the request is accepted and Accept = false otherwise.

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

# Device Connection

Connection of a device to a Mesh Account combines synchronous and asynchronous elements and therefore uses a combination of Mesh Service Protocol and Mesh Messaging interactions.

Three connection mechanisms are currently defined. All three of which offer strong mutual authentication.

* Device Authenticated
* Pin Authenticated
* EARL Connection Mode

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

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

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

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

## Device Authenticated

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

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

## PIN Authenticated

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

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

## EARL connection mode

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

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

# Protocol Schema

<include=..\Generated\ProtocolSchema.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"/>.