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

The Mesh specifies two separate types of protocol interactions:

Mesh Service Protocol

A synchronous protocol supporting interactions between devices and a Mesh Service Host and between Mesh Service hosts.

Mesh Messaging Protocol

An asynchronous protocol that supports interactions between devices connected to the same account and between accounts.

The Mesh Messaging Protocol uses the Mesh Service Protocol as transport. The Mesh Service Protocol in turn makes use of Reliable UDP Datagram (RUD) <norm="draft-hallambaker-mesh-rud"/> for framing and authentication of individual requests and responses. These RUS packets are in turn exchanged over either HTTPS (i.e. a Web Service) or directly over UDP.

<figuresvg="../Images/ProtocolLayering.svg"> Protocol Layering

Mesh Services MUST support the HTTPS binding and MAY support the UDP binding.

# 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

The protocol binding maps the abstract protocol definition specified in this document to the network protocol format.

* Discovery of network services.
* Construction of the payload data by serializing request and response messages.
* Authentication of the payload data.
* Confidentiality controls to protect against traffic analysis

Currently only one protocol binding is specified: JSON-BCD Application Binding <norm="draft-hallambaker-jsonbcd"/> over Reliable User Datagram (RUD) <norm="draft-hallambaker-mesh-rud"/>.

**JSON-BCD Application Binding** specifies the means by which data types such as 'integer' and 'datetime' etc. given in this document are serialized using JSON/JSON-B encoding.

**Reliable User Datagram** offers a presentation layer over a choice of HTTP or UDP transport.

# Mesh Service Operations

The Mesh Service operations are divided into the following functional groups:

Service Description

Describes the service.

Account Management

Operations used to create, reclaim, and delete accounts.

Persistence Store Management

Operations used to synchronize persistence store data across connected devices. [May be replaced in a future revision]

Device Connection

Operations used by devices requesting connection to the account.

Publication

Operations allowing a watched document to be posted to the service and claims made on the document returned to a device.

Cryptographic

Cryptographic operations, including threshold operations performed by the service.

Messaging

Exchange of messages between Mesh Services.

## Service Description

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

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

The current revision of the specification is designed for small scale deployments in which the service is provided by a single host. The approach will require revision in future versions to fully support a service being provided by multiple hosts with accounts being transferred between the hosts to allow balancing of load.

## Account Management

There are three account management operations:

BindAccount

Create an account bound to a service address.

UnbindAccount

Delete an account bound to a service address

RecoverAccount

[TBS] Reclaim an account using a recovered primary secret.

The BindAccount operation is used to create User and Group accounts. Currently, these account types are distinct. This may change in future releases.

### Bind Account

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

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.

The request payload contains all the information needed to create the account:

* The account address
* The account profile

Since there is no Access Catalog until the account is created, the Bind Account request and subsequent requests used to initialize the access catalog for the account MUST be authenticated by the Account Authentication key.

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

#### Bind Group Account

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

#### Account Recovery

Should all the administration devices be lost, an account MAY be recovered by the process of recovering the profile master secret and using it to access the account through the account authentication key.

### Unbind Account

An account registration is deleted using the UnbindAccount transaction.

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

#### Account Transfer

Should a user wish to transfer their account to a new service provider, they first use the Bind Account operation to bind the account to the new service provider, then populate the account entry at the new account using the account authentication key.

Only after the new account binding has been completed and is ready for use, is the unbind operation used to delete the account entry at the old service provider.

Future versions of the protocol will elaborate on this mechanism so that the change of address can be signaled to connected devices and parties sending messages to the account.

### Account Recovery and Transfer.

Account recovery is necessary in the case that user has lost control of every administration device connected to the account and must re-create the account profile and bind a new set of administrative devices. Account transfer is the process of unbinding an account from one service and rebinding it to a new one.

These capabilities are both critical to the long term success of the Mesh but have been deleted from the current revision of the specification as their implementation is interdependent on the architecture of the callsign registry.

<include=..\Examples\ProtocolAccountRecover.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 32 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.

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

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

Future versions of the protocol may support optional filtering criteria so that the service only returns objects matching specific criteria and/or only return certain parts of the selected messages.

### 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 allows Mesh stores to provide ACID (Atomicity, Consistency, Isolation, Durability) properties to the applications they serve.

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.

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>

## Device Connection

In order to support the wide range of affordances supported by devices, four device connection interactions are currently specified. The use of these mechanisms is described in <norm="draft-hallambaker-mesh-architecture"/> and the interactions themselves are described in section ??? following.

Device connection operations are always issued by a device requesting connection to a Mesh account and must therefore be authenticated under the device profile rather than the account profile. Two device connection operations are currently defined:

Connect

Requests connection to the account.

Complete

Polls for completion of a connection request.

Since the second operation is merely polling for completion of the transaction requested by the first, it is likely that these will be combined in a future revision of the specification.

### Connect

If the connection request is initiated by the device being connected, the device constructs a RequestConnection message which is posted to the Mesh Service using the Connect operation.

If the Connect operation is accepted (i.e. the service determines it is not abuse), the service constructs an AcknowledgeConnection message which is forwarded to the inbound spool of the account to which connection is requested. The requesting device receives a copy of the AcknowledgeConnection message and the profile of the account it is requesting connection to.

As described in the following section, the AcknowledgeConnection message contains the request details presented by the device and a nonce value generated by the service. This nonce value is used to compute the witness value that will be used for mutual authentication of the device and account.

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

### Complete

The complete operation is used to complete the binding of a device to the account regardless of whether the operation is initiated by the administration device or the connecting device.

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

## Publication

[Future: Consider eliminating this mechanism entirely and instead using messaging flows. The means of achieving this should become better apparent when the problem of publishing large messages via a pull mechanism is considered.]

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. There are two operations:

Claim

Post a claim to a published document

PollClaim

Check to see if a claim has been posted.

Content is published by appending an entry to an account's Publication catalog by means of a Transact operation. 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 catalog 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 Transaction

The claim transaction is used to post a claim to a document published by means of an EARL. The claim interaction is used in the Static QR Code connection interaction but MAY be used for other purposes as required by Mesh applications.

A claim is made by sending a ClaimRequest message to the service to which the publication is posted. The service responds with a ClaimRespose message specifying the success or failure of the claim.

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

### PollClaim Transaction

The PollClaim transaction is used to discover if a claim has been posted to a published document.

When an authenticated, authorized request is made, the service responds with the latest claim posted to the publication.

<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 Access catalog, the request MUST meet the authentication criteria specified by the catalog entry. These typically include the request being authenticated by a specific key.

Key Agreement

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

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

**Future**: Currently, the access catalog is encrypted under the service encryption key. It would be better to encrypt the catalog under an encryption key specified by the service during the process of account binding. This would allow a service to assign a unique encryption key to each account and limit access to that key to the hosts servicing that specific account.

### Generate Key Shares

Generation of threshold key shares is planned but not currently supported.

### Threshold Sign

Threshold signature is planned but not currently supported.

## 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/ArchFourCorner.svg">The Mesh Four Corner Messaging Model

The Post transaction is only used to exchange messages between services. The client sends and receives messages through interactions with the outbound and inbound spools of the account.

### Sender.

To send a message, the client creates the Mesh Message structure, encapsulates it in a DARE Message and appends the message to the Outbound spool of the account using the Transact operation..

The DARE Message MUST be signed under the account signature key.

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.

### Outbound Service

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>

After the message has been sent, the service updates the message status on the outbound spool.

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 completed in less than a second, the number of threads allocated to sending outbound messages might be increased.

### Inbound Service

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.

### Recipient

Messages are received by synchronizing the outbound spool.

# Access Control

[This section to be expanded in future drafts]

Access control is effected through the usual division of authentication and authorization.

Authentication of operation requests is performed by the RUD layer <norm="draft-hallambaker-mesh-rud"/> .

## Direct authorization

Any request authenticated under the profile authentication key is authorized to perform any account operation without restriction.

## Access Catalog authentication

If the authentication key presented has a matching Access Catalog entry, the device is authorized to perform operations as specified in that entry.

# Message Interactions

Message interactions are asynchronous interactions that occur between devices connected to the same account or 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.

## Message PIN Interaction

The Message PIN Interaction is used to register and validate PIN codes used to authenticate certain transactions. This interaction allows a PIN code issued by one device to be consumed by another allowing for greater convenience in managing devices or contact exchange.

For example, Alice might delegate the PIN code issue privilege to her mobile device without delegating the administration privilege to that device. This would allow Alice to use her mobile device to initiate the connection of a large number of devices to her Mesh as her house is being built and approve them later using her administrative device.

Use of the Message PIN interaction is optional. An application that issues a PIN code to authenticate a message MAY store the PIN value within the application without persisting it to external storage.

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.

### Example

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

## Completion Interaction

Completion messages are dummy messages that are added to a Mesh Spool to mark a 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 message will be removed if the sequence is subsequently purged.

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

The completion message is added to the spool in the same upload transaction that adds the device to the device catalog. This ensures that both operations occur or neither occurs.

## Contact Exchange Interaction

The contact exchange interaction 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 a registry 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.

The communication of unsolicited information afforded by the contact exchange interaction is deliberately limited so that a majority of users can accept contact exchange requests without prior authorization. It is however likely that some users will receive a considerable volume of requests forcing them to require contact requests be authorized through some form of third party accreditation.

### 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

Contact exchange requests MAY be authenticated by a PIN code. Initial contact exchange requests SHOULD include a PIN code value that can be used to authenticate a response (if given). PIN codes MAY also be exchanged out of band.

A MessageContact authenticated by means of a PIN code is authenticated as described in the PIN Interaction section above.

### 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.

## Group Invitation

The GroupInvitation interaction is used to invite a recipient to join a Mesh Group. The interaction is essentially a form of contact exchange except that a sender SHOULD NOT send group invitations unless there is an existing relationship. Thus the 'first trust' issues intrinsic to the contact exchange interaction do not apply.

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 Interaction

The confirmation interaction 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 Interactions

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.

Four connection interactions are currently defined support connection of devices with different affordances:

Witness Authenticated

For connecting devices that provide data entry and display affordances and are connected to a network. The account the device is to be connected to is entered into the device which displays a witness code. This code is then compared with a code displayed on the administration device to authenticate the request, after which both devices can complete the interaction.

PIN Authenticated

A variation of the Witness Authenticated interaction in which the connection process is initiated by creating a PIN value which is communicated to the device by some out of band means and used to authenticate the connection request.

Dynamic QR Code (PIN) Authenticated

For connecting devices that provide a camera affordance. The user sets the administration device into 'add device' mode, causing a QR code to be displayed. The QR code is scanned by the device being connected after which both devices can complete the interaction. Implementation of this mechanism is identical to the PIN authenticated scheme except that the PIN code is presented to the connecting device by means of a QR code.

Preconfigured (Static QR Code Authenticated)

For connecting devices that have been preconfigured with a device profile identified by means of a QR Code containing an EARL. The QR code is scanned by the administration device after which both devices can complete the interaction.

Each of these interactions provide strong mutual authentication with minimal user effort.

The witness authenticated connection interaction is intended for use in cases in which the device is already connected to a network. The QR code interactions are intended to provide support for acquisition of networking capabilities as part of the connection process. These functions are not currently specified. The Static QR Code Authenticated interaction is intended to support Internet of Things (IoT) devices which provide minimal interaction affordances.

In each case, the objectives of the device connection interaction are the same:

* Mutually authenticate the onboarding device and the Mesh such that the connection interaction only completes if both sides acquire the authentic profile of the other.
* To provision the onboarding device with the Mesh ProfileAccount, and an ActivationDevice and ConnectionDevice record allowing the device to interact as a member of the Mesh with the set of rights specified by the user.
* To create a CataloguedDevice record and append it to the Device catalog of the account to allow the device to be managed within that account.
* (optional) to acquire networking capabilities to allow the above to be completed.

The connection of the device to the Mesh Account is achieved through the creation of the ActivationDevice, ConnectionDevice and CataloguedDevice records described in <norm="draft-hallambaker-mesh-schema"/>. These are created by the administration device in the third phase of each of the connection interactions described below and acquired by the onboarding device in the fourth phase.

## Witness/PIN Authenticated

The witness authenticated, PIN authenticated, and Dynamic QR code interactions all follow a common interaction pattern.

The Dynamic QR Code (PIN) Authenticated interaction comprises four phases as follows:

Phase 1: Issue of PIN credential (PIN and Dynamic QR code only)

A PIN code is created and registered with the PIN Registration interaction described earlier and transmitted to the user by an out of band communication. In the case of the Dynamic QR code interaction, this is a QR code that is scanned by the connecting device.

Phase 2: Onboarding Device Request to Service

The onboarding device creates a RequestConnect message. In the PIN authenticated and Dynamic QR Code interactions, the RequestConnect is authenticated by the Device Authentication key and the PIN issued earlier. In the Witness Authenticated interaction, it is authenticated by the Device Authentication key alone.

The onboarding device presents the RequestConnect message to the service by means of a Connect operation to the service servicing the account. This results in the exchange of the account and device profiles and the computation of a witness value from the two profile fingerprints and two nonce values specified by the onboarding device and the service. An AcknowledgeConnection message is posted to the Inbound spool of the account and returned to the connecting device.

Phase 3: Administration Device Acceptance

The account holder authenticates RequestConnect message and uses an administrative device to accept or reject the connection request.

If the RequestConnect message has been authenticated by a PIN code, the connection request can be accepted automatically without additional user interaction.

Phase 4: Onboarding Device Completion

The onboarding device periodically polls the service for acceptance of the request by the administration device using the Complete transaction.

The use of the PIN code to authenticate the request message is shown in $$$$.

The PIN code MAY be presented to the onboarding device in any format accepted by the device. Administration MAY support presentation of the account address PIN code as a URI code. Administration devices SHOULD support presentation of the account address PIN code as a QR code containing the corresponding URI.

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

## Preconfigured (Static QR Code)

The preconfigured device connection interaction is used to connect devices that lack affordances such as a display or a keyboard. It is also known as the static QR code interaction because a static QR code printed on the device itself is used to connect it to a user's account.

**Future**: Note that this interaction is likely to be changed substantially in future revisions of the specification and the Claim/PollClaim mechanism removed and replaced with a messaging based approach.

The interaction has five phases:

Phase 1: Preconfiguration

The device to be onboarded is preconfigured with a ProfileDevice and private key information and a DeviceDescription posted to a publication service. This process is typically performed during manufacture. An EARL providing the ability to locate and decrypt the description is printed on the device itself as a QR code.

Phase 2: Device description acquisition

The administration device acquiring the onboarding device scans the QR code on the device and uses this information to obtain the device description by means of a Claim operation described above as described in the Device Description.

Phase 3: Administration Device Acceptance

This phase is performed in the same manner as the Dynamic QR Code (PIN) Authenticated interaction except that the administration device MAY advise the device that a connection request is being made by additional means described in the device description (e.g. WiFi, Bluetooth).

Phase 4: Poll Claim Notification

When connected to a network, the preconfigured device periodically attempts to poll the connection sources specified to find out if there is a pending request. If a connection request is posted, the device decrypts it to allow it to complete the connection process.

Phase 5: Onboarding Device Completion

This phase is performed in the same manner as the Dynamic QR Code (PIN) Authenticated interaction except that the administration device requires notice that of the pending connection request.

The main differences between this connection interaction and the witness/PIN connection interactions are that the device is preconfigured with the device profile at the time of manufacture and the onboarding device MAY be acquiring network configuration information during the connection process.

<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"/>.