Abstract

The Mathematical Mesh 'The Mesh' is an end-to-end secure infrastructure that facilitates the exchange of configuration and credential data between multiple user devices. The core protocols of the Mesh are described with examples of common use cases and reference data.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 23, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
# Table of Contents

1. Introduction ........................................... 4  
2. Definitions ............................................ 4  
    2.1. Requirements Language ............................... 4  
3. Architecture ......................................... 5  
    3.1. Data Model ......................................... 5  
    3.1.1. First Class Object ............................. 5  
    3.1.2. Profile ......................................... 5  
    3.1.3. Record .......................................... 5  
    3.1.4. Transaction .................................... 5  
    3.2. Profile Types ...................................... 5  
    3.3. Master Profile .................................... 5  
    3.4. Personal Profile .................................. 6  
    3.5. Device Profile .................................... 6  
    3.6. Application Profile ............................... 6  
4. Cryptographic Data Objects ............................. 7  
    4.1. Public Key Objects ................................ 7  
    4.1.1. Structure: PublicKey ........................... 7  
    4.2. JOSE Signature Objects ............................ 7  
    4.2.1. Structure: SignedData .......................... 7  
    4.3. JOSE Encryption Objects ........................... 7  
    4.3.1. Structure: EncryptedData ........................ 7  
5. Mesh Profile Objects .................................. 8  
    5.1. Base Profile Objects ............................... 8  
    5.1.1. Structure: Entry ................................ 8  
    5.1.2. Structure: SignedProfile ....................... 8  
    5.1.3. Structure: Profile ............................... 8  
    5.2. Device Profile Objects ............................ 9  
    5.2.1. Structure: SignedDeviceProfile ................. 9  
    5.2.2. Structure: DeviceProfile ....................... 9  
    5.2.3. Structure: DevicePrivateProfile ............... 10  
    5.3. Master Profile Objects ............................ 10  
    5.3.1. Structure: SignedMasterProfile ................. 10  
    5.3.2. Structure: MasterProfile ....................... 11  
    5.4. Personal Profile Objects .......................... 11  
    5.4.1. Structure: SignedPersonalProfile .............. 11  
    5.4.2. Structure: PersonalProfile ..................... 12  
    5.5. Application Profile Objects ...................... 12  
    5.5.1. Structure: SignedApplicationProfile .......... 12  
    5.5.2. Structure: EncryptedProfile .................... 12  
    5.5.3. Structure: ApplicationProfile .................. 13  
    5.5.4. Structure: ApplicationProfileEntry ........... 13  
    5.6. Common Application Objects ...................... 14  
    5.6.1. Structure: Connection ........................... 14  
    5.7. Password Application Profile Objects ............ 15  
    5.7.1. Structure: PasswordProfile ..................... 15  
    5.7.2. Structure: PasswordProfilePrivate ............. 15
5.7.3. Structure: PasswordEntry

5.8. Mail Application Profile Objects

5.8.1. Structure: MailProfile

5.8.2. Structure: MailProfilePrivate

5.9. Network Application Profile Objects

5.9.1. Structure: NetworkProfile

5.9.2. Structure: NetworkProfilePrivate

5.10. Key Escrow Objects

5.10.1. Structure: EscrowEntry

5.10.2. Structure: OfflineEscrowEntry

5.10.3. Structure: OnlineEscrowEntry

5.10.4. Structure: EscrowedKeySet

6. Portal Connection

6.1. Connection Request and Response Structures

6.1.1. Structure: ConnectionRequest

6.1.2. Structure: SignedConnectionRequest

6.1.3. Structure: ConnectionResult

6.1.4. Structure: SignedConnectionResult

7. Mesh Portal Service Reference

7.1. Request Messages

7.1.1. Message: MeshRequest

7.1.2. Response Messages

7.2.1. Message: MeshResponse

7.2.2. Successful Response Codes

7.2.3. Warning Response Codes

7.2.4. Error Response Codes

7.2.5. Failure Response Codes

7.3. Imported Objects

7.4. Common Structures

7.4.1. Structure: Version

7.4.2. Structure: Encoding

7.4.3. Structure: KeyValue

7.4.4. Structure: SearchConstraints

7.5. Transaction: Hello

7.5.1. Message: HelloRequest

7.5.2. Message: HelloResponse

7.6. Transaction: ValidateAccount

7.6.1. Message: ValidateRequest

7.6.2. Message: ValidateResponse

7.7. Transaction: CreateAccount

7.7.1. Message: CreateRequest

7.7.2. Message: CreateResponse

7.8. Transaction: Get

7.8.1. Message: GetRequest

7.8.2. Message: GetResponse

7.9. Transaction: Publish

7.9.1. Message: PublishRequest

7.9.2. Message: PublishResponse
1. Introduction

NB: The reference material in this document is generated from the schema used to derive the source code. The tool used to create this material has not been optimized to produce output for the IETF documentation format at this time. Consequently the formatting is currently sub-optimal.

2. Definitions

2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].
3. Architecture

3.1. Data Model

3.1.1. First Class Object

3.1.2. Profile

A profile is a first class object. It has a globally unique identifier that provides an unambiguous reference to the profile in any situation.

3.1.3. Record

A record describes the state of an object at the completion of a specific Transaction.

3.1.4. Transaction

A transaction is an event in which the state of an object changes. Every transaction has a globally unique transaction identifier. Transaction identifiers are issued in a monotonic sequence such that a transaction that completes at time t1 will always have a lower transaction identifier than one that begins at time t2 where t2 > t1.

3.2. Profile Types

- Master Profile
- Personal Profile
- Application Profile
- Device Profile

3.3. Master Profile

The master profile contains the axioms of trust for a Mesh user.

Identifier: "Master" + UDF Fingerprint of the Master Signing Key

Signature: Master Signing Key  The key used to sign the profile MUST be MasterSigningKey
Property: Master Signing Key  The Master Signing key is the ultimate trust axiom for the Master Profile.

Property: Master Escrow Keys

Property: Online Signature Keys

3.4. Personal Profile

Identifier: UDF Fingerprint of the Master Signing Key

Signature: Online Signature Key  The key used to sign the profile MUST be a member of MasterProfile/OnlineSignatureKeys

Property: Master Profile  The Master Profile that this personal profile is an instance of.

Property: Devices

Property: Applications  A list of application profile entries specifying which application profiles are attached to the personal profile

3.5. Device Profile

Identifier: UDF Fingerprint of the Device Signing Key

Signature: Device Signing Key  The key used to sign the profile MUST be MasterSigningKey

Property: Device Signing Key  The Master Signing key is the ultimate trust axiom for the Master Profile.

Property: Device Encryption Key

Property: Device Authentication Key

3.6. Application Profile

Identifier: Randomly chosen

Property: Encrypted Data
4. Cryptographic Data Objects

4.1. Public Key Objects

4.1.1. Structure: PublicKey

Container for public key pair data

UDF: String (Optional)

UDF fingerprint of the key

X509Certificate: Binary (Optional)

List of X.509 Certificates

X509Chain: Binary [0..Many]

X.509 Certificate chain.

X509CSR: Binary (Optional)

X.509 Certificate Signing Request.

4.2. JOSE Signature Objects

4.2.1. Structure: SignedData

Container for JOSE signed data and related attributes.

Data: Binary (Optional)

4.3. JOSE Encryption Objects

4.3.1. Structure: EncryptedData

Container for JOSE encrypted data and related attributes.

Data: Binary (Optional)
5. Mesh Profile Objects

5.1. Base Profile Objects

5.1.1. Structure: Entry

Base class for all Mesh Profile objects.

Identifier: String (Optional)

Globally unique identifier that remains constant for the lifetime of the entry.

5.1.2. Structure: SignedProfile

Contains a signed profile entry

SignedData: JoseWebSignature (Optional)

The signed profile.

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

5.1.3. Structure: Profile

Parent class from which all profile types are derived

Names: String [0..Many]

Fingerprints of index terms for profile retrieval. The use of the fingerprint of the name rather than the name itself is a
precaution against enumeration attacks and other forms of abuse.

Updated: DateTime (Optional)

The time instant the profile was last modified.

NotaryToken: String (Optional)

A Uniform Notary Token providing evidence that a signature was performed after the notary token was created.

5.2. Device Profile Objects

5.2.1. Structure: SignedDeviceProfile

* Inherits: SignedProfile

Contains a signed device profile

5.2.2. Structure: DeviceProfile

* Inherits: Profile

Describes a mesh device.

Description: String (Optional)

Description of the device

DeviceSignatureKey: PublicKey (Optional)

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

Key used to authenticate requests made by the device.

DeviceEncryptionKey: PublicKey (Optional)

Key used to pass encrypted data to the device such as a DeviceUseEntry

5.2.3. Structure: DevicePrivateProfile

Private portion of device encryption profile.

DeviceSignatureKey: Key (Optional)

Private portion of the DeviceSignatureKey

DeviceAuthenticationKey: Key (Optional)

Private portion of the DeviceAuthenticationKey

DeviceEncryptionKey: Key (Optional)

Private portion of the DeviceEncryptionKey

5.3. Master Profile Objects

5.3.1. Structure: SignedMasterProfile

* Inherits: SignedProfile

Contains a signed Personal master profile

[None]
5.3.2. Structure: MasterProfile

* Inherits: Profile

Describes the long term parameters associated with a personal profile.

MasterSignatureKey: PublicKey (Optional)

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

MasterEscrowKeys: PublicKey [0..Many]

A Personal Profile MAY contain one or more PMEK keys to enable escrow of private keys used for stored data.

OnlineSignatureKeys: PublicKey [0..Many]

A Personal profile contains at least one POSK which is used to sign device administration application profiles.

5.4. Personal Profile Objects

5.4.1. Structure: SignedPersonalProfile

* Inherits: SignedProfile

Contains a signed Personal current profile

[None]
5.4.2. Structure: PersonalProfile

- Inherits: Profile

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

SignedMasterProfile: SignedMasterProfile (Optional)

The corresponding master profile. The profile MUST be signed by the PMSK.

Devices: SignedDeviceProfile [0..Many]

The set of device profiles connected to the profile. The profile MUST be signed by the DSK in the profile.

Applications: ApplicationProfileEntry [0..Many]

Application profiles connected to this profile.

5.5. Application Profile Objects

5.5.1. Structure: SignedApplicationProfile

- Inherits: SignedProfile

Contains a signed device profile

[None]

5.5.2. Structure: EncryptedProfile

- Inherits: Entry

Contains an encrypted profile entry
5.5.3. Structure: ApplicationProfile

- Inherits: Profile

Parent class from which all application profiles inherit.

5.5.4. Structure: ApplicationProfileEntry

Identifier: String (Optional)

The unique identifier of the application

Type: String (Optional)

The application type

Friendly: String (Optional)

Optional friendly name identifying the application.

SignID: String [0..Many]

List of devices authorized to sign application profiles

DecryptID: String [0..Many]
5.6. Common Application Objects

5.6.1. Structure: Connection

Describes network connection parameters for an application

ServiceName: String (Optional)

DNS address of the server

Port: Integer (Optional)

TCP/UDP Port number

Prefix: String (Optional)

DNS service prefix as described in [RFC6335]

Security: String [0..Many]

Describes the security mode to use. Valid choices are Direct/Upgrade/None

UserName: String (Optional)

Username to present to the service for authentication

Password: String (Optional)

Password to present to the service for authentication

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

Authentication: String (Optional)

List of the supported/acceptable authentication mechanisms, preferred mechanism first.

TimeOut: Integer (Optional)

Service timeout in seconds.

Polling: Boolean (Optional)

If set, the client should poll the specified service intermittently for updates.

5.7. Password Application Profile Objects

5.7.1. Structure: PasswordProfile

* Inherits: ApplicationProfile

Stores usernames and passwords

[None]

5.7.2. Structure: PasswordProfilePrivate

AutoGenerate: Boolean (Optional)

If true, a client MAY offer to automatically generate strong (i.e. not memorable) passwords for a user. A user would not normally want to use this feature unless they have access to Mesh password management on every device they use to browse the Web.

Entries: PasswordEntry [0..Many]
A list of password credential entries.

NeverAsk: String [0..Many]

A list of domain names of sites for which clients MUST NOT ask to store passwords for.

5.7.3. Structure: PasswordEntry

Username password entry for a single site

Sites: String [0..Many]

DNS name of site *.example.com matches www.example.com etc.

Username: String (Optional)

Case sensitive username

Password: String (Optional)

Case sensitive password.

5.8. Mail Application Profile Objects

5.8.1. Structure: MailProfile

* Inherits: ApplicationProfile

Public profile describes mail receipt policy. Private describes sending policy

EncryptionPGP: PublicKey (Optional)

The current OpenPGP encryption key
EncryptionSMIME: PublicKey (Optional)

The current S/MIME encryption key

5.8.2. Structure: MailProfilePrivate

Describes a mail account configuration

Private profile contains connection settings for the inbound and outbound mail server(s) and cryptographic private keys. Public profile may contain security policy information for the sender.

EmailAddress: String (Optional)

The RFC822 Email address. [e.g. "alice@example.com"]

ReplyToAddress: String (Optional)

The RFC822 Reply toEmail address. [e.g. "alice@example.com"]

When set, allows a sender to tell the receiver that replies to this account should be directed to this address.

DisplayName: String (Optional)

The Display Name. [e.g. "Alice Example"]

AccountName: String (Optional)

The Account Name for display to the app user [e.g. "Work Account"]

Inbound: Connection [0..Many]

The Inbound Mail Connection(s). This is typically IMAP4 or POP3
If multiple connections are specified, the order in the sequence indicates the preference order.

Outbound: Connection [0..Many]

The Outbound Mail Connection(s). This is typically SMTP/SUBMIT

If multiple connections are specified, the order in the sequence indicates the preference order.

Sign: PublicKey [0..Many]

The public keypair(s) for signing and decrypting email.

If multiple public keys are specified, the order indicates preference.

Encrypt: PublicKey [0..Many]

The public keypairs for encrypting and decrypting email.

If multiple public keys are specified, the order indicates preference.

5.9. Network Application Profile Objects

5.9.1. Structure: NetworkProfile

  * Inherits: ApplicationProfile

Describes the network profile to follow

[None]

5.9.2. Structure: NetworkProfilePrivate

Describes the network profile to follow

Sites: String [0..Many]
DNS name of sites to which profile applies *.example.com matches www.example.com etc.

DNS: Connection [0..Many]

DNS Resolution Services
Prefix: String [0..Many]

DNS prefixes to search
CTL: Binary (Optional)

Certificate Trust List giving WebPKI roots to trust
WebPKI: String [0..Many]

List of UDF fingerprints of keys making up the trust roots to be accepted for Web PKI purposes.

5.10. Key Escrow Objects

5.10.1. Structure: EscrowEntry

  * Inherits: Entry
  
  Contains escrowed data
  
  EncryptedData: JoseWebEncryption (Optional)

5.10.2. Structure: OfflineEscrowEntry

  * Inherits: EscrowEntry
  
  Contains data escrowed using the offline escrow mechanism.
5.10.3. Structure: OnlineEscrowEntry

* Inherits: EscrowEntry

Contains data escrowed using the online escrow mechanism.

5.10.4. Structure: EscrowedKeySet

A set of escrowed keys.

PrivateKeys: Key [0..Many]

The escrowed keys.

6. Portal Connection

6.1. Connection Request and Response Structures

6.1.1. Structure: ConnectionRequest

Describes a connection request.

ParentUDF: String (Optional)

UDF of Mesh Profile to which connection is requested.

Device: SignedDeviceProfile (Optional)

The Device profile to be connected

6.1.2. Structure: SignedConnectionRequest

* Inherits: SignedProfile
Contains a ConnectionRequest signed by the corresponding device signature key.

[None]

6.1.3. Structure: ConnectionResult

Describes the result of a connection request.

- * Inherits: ConnectionRequest
  
  Result: String (Optional)

  The result of the connection request. Valid responses are: Accepted, Refused, Query.

6.1.4. Structure: SignedConnectionResult

- * Inherits: SignedProfile
  
  Contains a signed connection result

[None]

7. Mesh Portal Service Reference

SRV Prefix:

_mmm._tcp

HTTP Well Known Service Prefix:

/.well-known/mmm

Every Mesh Portal Service transaction consists of exactly one request followed by exactly one response. Mesh Service transactions MAY cause modification of the data stored in the Mesh Portal or the Mesh itself but do not cause changes to the connection state. The
protocol itself is thus idempotent. There is no set sequence in which operations are required to be performed. It is not necessary to perform a Hello transaction prior to a ValidateAccount, Publish or any other transaction.

7.1. Request Messages

A Mesh Portal Service request consists of a payload object that inherits from the MeshRequest class. When using the HTTP binding, the request MUST specify the portal DNS address in the HTTP Host field.

7.1.1. Message: MeshRequest

Base class for all request messages.

Portal: String (Optional)

Name of the Mesh Portal Service to which the request is directed.

7.2. Response Messages

A Mesh Portal Service response consists of a payload object that inherits from the MeshResponse class. When using the HTTP binding, the response SHOULD report the Status response code in the HTTP response message. However the response code returned in the payload object MUST always be considered authoritative.

7.2.1. Message: MeshResponse

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

A service MAY return either the response message specified for that transaction or any parent of that message. Thus the MeshResponse message MAY be returned in response to any request.

Status: Integer (Optional)

Status return code. The SMTP/HTTP scheme of 2xx = Success, 3xx = incomplete, 4xx = failure is followed.

StatusDescription: String (Optional)
Text description of the status return code for debugging and log file use.

7.2.2. Successful Response Codes

The following response codes are returned when a transaction has completed successfully.

[201] SuccessOK

Operation completed successfully

[201] SuccessCreated

Operation completed successfully, new data item created

[202] SuccessUpdated

Operation completed successfully, data item was updated

7.2.3. Warning Response Codes

The following response codes are returned when a transaction did not complete because the target service has been redirected.

In the case that a redirect code is returned, the StatusDescription field contains the URI of the new service. Note however that the redirect location indicated in a status response might be incorrect or even malicious and cannot be considered trustworthy without appropriate authentication.

[303] RedirectPermanent

Service has been permanently moved

[307] RedirectTemporary

Service has been temporarily moved
7.2.4. Error Response Codes

A response code in the range 400-499 is returned when the service was able to process the transaction but the transaction resulted in an error.

[401] ClientUnauthorized

Client is not authorized to perform specified request

[404] NotFound

The requested object could not be found.

[409] AlreadyExists

The requested object already exists.

7.2.5. Failure Response Codes

A response code in the range 500-599 is returned when the service was unable to process the transaction but the transaction due to an internal failure.

[500] ServerInternal

An internal error occurred at the server

[503] ServerOverload

The server cannot handle the request as it is overloaded

7.3. Imported Objects

The Mesh Service protocol makes use of JSON objects defined in the JOSE Signature and Encryption specifications.
7.4. Common Structures

The following common structures are used in the protocol messages:

7.4.1. Structure: Version

Describes a protocol version.

Major: Integer (Optional)

Major version number of the service protocol. A higher

Minor: Integer (Optional)

Minor version number of the service protocol.

Encodings: Encoding [0..Many]

Enumerates alternative encodings (e.g. ASN.1, XML, JSON-B)
supported by the service. If no encodings are specified, the JSON
encoding is assumed.

URI: String [0..Many]

The preferred URI for this service. This MAY be used to effect a
redirect in the case that a service moves.

7.4.2. Structure: Encoding

Describes a message content encoding.

ID: String [0..Many]

The IANA encoding name

Dictionary: String [0..Many]
For encodings that employ a named dictionary for tag or data compression, the name of the dictionary as defined by that encoding scheme.

7.4.3. Structure: KeyValue

Describes a Key/Value structure used to make queries for records matching one or more selection criteria.

Key: String (Optional)

The data retrieval key.

Value: String (Optional)

The data value to match.

7.4.4. Structure: SearchConstraints

Specifies constraints to be applied to a search result. These allow a client to limit the number of records returned, the quantity of data returned, the earliest and latest data returned, etc.

NotBefore: DateTime (Optional)

Only data published on or after the specified time instant is requested.

Before: DateTime (Optional)

Only data published before the specified time instant is requested. This excludes data published at the specified time instant.

MaxEntries: Integer (Optional)

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

Maximum number of data bytes to return.

PageKey: String (Optional)

Specifies a page key returned in a previous search operation in which the number of responses exceeded the specified bounds.

When a page key is specified, all the other search parameters except for MaxEntries and MaxBytes are ignored and the service returns the next set of data responding to the earlier query.

7.5. Transaction: Hello

Request: HelloRequest

Response: HelloResponse

Report service and version information.

The Hello transaction provides a means of determining which protocol versions, message encodings and transport protocols are supported by the service.

7.5.1. Message: HelloRequest

o

  * Inherits: MeshRequest

[None]

7.5.2. Message: HelloResponse

Always reports success. Describes the configuration of the Mesh portal service.

o

  * Inherits: MeshResponse

Version: Version (Optional)
Enumerates the protocol versions supported

Alternates: Version [0..Many]

Enumerates alternate protocol version(s) supported

7.6. Transaction: ValidateAccount

Request: ValidateRequest
Response: ValidateResponse

Request validation of a proposed name for a new account.
For validation of a user’s account name during profile creation.

7.6.1. Message: ValidateRequest

* Inherits: MeshRequest

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

Account: String (Optional)

Account name requested

Reserve: Boolean (Optional)

If true, request a reservation for the specified account name.
Note that the service is not obliged to honor reservation requests.

Language: String [0..Many]

List of ISO language codes in order of preference. For creating explanatory text.
7.6.2. Message: ValidateResponse

- Inherits: MeshResponse

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

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

Valid: Boolean (Optional)

If true, the specified account identifier is acceptable. If false, the account identifier is rejected.

Minimum: Integer (Optional)

Specifies the minimum length of an account name.

Maximum: Integer (Optional)

Specifies the maximum length of an account name.

InvalidCharacters: String (Optional)

A list of characters that the service does not accept in account names. The list of characters MAY not be exhaustive but SHOULD include any illegal characters in the proposed account name.

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

7.7. Transaction: CreateAccount

Request: CreateRequest
Response: CreateResponse

Request creation of a new portal account.

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

7.7.1. Message: CreateRequest

Request creation of a new portal account. The request specifies the requested account identifier and the Mesh profile to be associated with the account.

* Inherits: MeshRequest

Account: String (Optional)

Account identifier requested.

7.7.2. Message: CreateResponse

* Inherits: MeshResponse

Reports the success or failure of a Create transaction.

[None]

7.8. Transaction: Get

Request: GetRequest
Response: GetResponse

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

Describes the Portal or Mesh data to be retrieved.

*  Inherits: MeshRequest

Identifier: String (Optional)

Lookup by profile ID

Account: String (Optional)

Lookup by Account ID

KeyValues: KeyValue [0..Many]

List of KeyValue pairs specifying the conditions to be met

SearchConstraints: SearchConstraints (Optional)

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

Multiple: Boolean (Optional)

If true return multiple responses if available

Full: Boolean (Optional)

If true, the client requests that the full Mesh data record be returned containing both the Mesh entry itself and the Mesh metadata that allows the date and time of the publication of the Mesh entry to be verified.
7.8.2. Message: GetResponse

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

* Inherits: MeshResponse

DataItems: DataItem [0..Many]

List of mesh data records matching the request.

PageKey: String (Optional)

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

7.9. Transaction: Publish

Request: PublishRequest

Response: PublishResponse

Publish a profile or key escrow entry to the mesh.

7.9.1. Message: PublishRequest

Requests publication of the specified Mesh entry.

* Inherits: MeshRequest

[None]

7.9.2. Message: PublishResponse

Reports the success or failure of a Publish transaction.

[None]
7.10. Transaction: Status

Request: StatusRequest

Response: StatusResponse

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

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

[Not currently implemented]

7.10.1. Message: StatusRequest

* Inherits: MeshRequest

Initiates a status transaction.

[None]

7.10.2. Message: StatusResponse

Reports the success or failure of a Status transaction.

* Inherits: MeshResponse

LastWriteTime: DateTime (Optional)

Time that the last write update was made to the Mesh

LastCheckpointTime: DateTime (Optional)

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

Time at which the next Mesh checkpoint should be calculated.

CheckpointValue: String (Optional)

Last checkpoint value.

7.11. Transaction: ConnectStart

Request: ConnectStartRequest

Response: ConnectStartResponse

Request connection of a new device to a mesh profile

7.11.1. Message: ConnectStartRequest

  *
  *
  * Inherits: MeshRequest

  Initial device connection request.

  SignedRequest: SignedConnectionRequest (Optional)

    Device connection request signed by the signature key of the device requesting connection.

    AccountID: String (Optional)

    Account identifier of account to which the device is requesting connection.

7.11.2. Message: ConnectStartResponse

  Reports the success or failure of a ConnectStart transaction.
7.12. Transaction: ConnectStatus

Request: ConnectStatusRequest

Response: ConnectStatusResponse

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

7.12.1. Message: ConnectStatusRequest

* Inherits: MeshRequest

[None]

AccountID: String (Optional)

Account identifier for which pending connection information is requested.

DeviceID: String (Optional)

Device identifier of device requesting status information.

7.12.2. Message: ConnectStatusResponse

Reports the success or failure of a ConnectStatus transaction.

* Inherits: MeshRequest

Result: SignedConnectionResult (Optional)

The signed ConnectionResult object.
7.13. Transaction: ConnectPending

Request: ConnectPendingRequest
Response: ConnectPendingResponse

Request a list of pending requests for an administration profile.

7.13.1. Message: ConnectPendingRequest

* Inherits: MeshRequest

Specify the criteria for pending requests.

AccountID: String (Optional)

The account identifier of the account for which pending connection requests are requested.

SearchConstraints: SearchConstraints (Optional)

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

7.13.2. Message: ConnectPendingResponse

Reports the success or failure of a ConnectPending transaction.

* Inherits: MeshRequest

Pending: SignedConnectionRequest [0..Many]

A list of pending requests satisfying the criteria set out in the request.

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

7.14. Transaction: ConnectComplete

Request: ConnectCompleteRequest

Response: ConnectCompleteResponse

Post response to a pending connection request.


Reports the success or failure of a ConnectComplete transaction.

* Inherits: MeshRequest

Result: SignedConnectionResult (Optional)

The connection result to be posted to the portal. The result MUST be signed by a valid administration key for the Mesh profile.

AccountID: String (Optional)

The account identifier to which the connection result is posted.


* Inherits: MeshRequest

Reports the success or failure of a ConnectComplete transaction.

[None]
7.15. Transaction: Transfer

Request: TransferRequest

Response: TransferResponse

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

[Not currently implemented]

7.15.1. Message: TransferRequest

* Inherits: MeshRequest

SearchConstraints: SearchConstraints (Optional)

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

7.15.2. Message: TransferResponse

* Inherits: MeshResponse

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

DataItems: DataItem [0..Many]

List of mesh data records matching the request.

PageKey: String (Optional)

If non-null, indicates that the number and/or size of the data records returned exceeds either the SearchConstraints specified in the request or internal server limits.
8. Mesh Portal Objects

The precise implementation of the portal service and the data structures representing state at the portal service are outside the scope of this specification.

The specification of the Mesh Portal objects given here is to enable future formal specification of the portal protocols by defining the state changes resulting from portal transactions.

8.1. Mesh Portal Log Entries

Like the Mesh itself, the state of the portal is tracked by an append only log. This log contains entries binding account identifiers to mesh profiles and lists of pending connections.

8.1.1. Structure: PortalEntry

Created: DateTime (Optional)

Time the pending item was created.

Modified: DateTime (Optional)

Time the pending item was last modified.

8.1.2. Structure: Account

Entry containing the UniqueID is Account[Name]--[Portal] Indexed by [Name], [UserProfileUDF] [Most recent open]

* Inherits: PortalEntry

AccountID: String (Optional)

Assigned account identifier, e.g. ‘alice@example.com’. Account names are not case sensitive.

UserProfileUDF: String (Optional)
Fingerprint of associated user profile

Status: String (Optional)

Status of the account, valid values are 'Open', 'Closed', 'Suspended'

8.1.3. Structure: AccountProfile

  *
  * Inherits: Account

Profile: SignedPersonalProfile (Optional)

The personal profile associated with the account.

8.1.4. Structure: ConnectionsPending

Object containing the list of currently pending device connection requests for the specified account. Unique-ID is ConnectionsPending-[UserProfileUDF]

  *
  * Inherits: Account

Requests: SignedConnectionRequest [0..Many]

List of pending requests

9. Security Considerations

  TBS

9.1. Confidentiality

9.2. Integrity
9.3. Service

10. IANA Considerations

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

11. Acknowledgements

12. Normative References


Author’s Address

Phillip Hallam-Baker
Comodo Group Inc.

Email: philliph@comodo.com