Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 4, 2020

P. Hallam-Baker
July 3, 2019

Mathematical Mesh 3.0 Part IV: Schema Reference
draft-hallambaker-mesh-schema-01

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.

This document is also available online at http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html [1] .

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

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

This document has two main sections. The first section presents examples of the Mesh assertions, catalog entry and messages in use. The second section contains the schema reference. All the material in both sections is generated from the Mesh reference implementation [draft-hallambaker-mesh-developer] .

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

2. Definitions

This section presents the related specifications and standard, the terms that are used as terms of art within the documents and the terms used as requirements language.

2.1. Requirements Language

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

2.2. Defined Terms

The terms of art used in this document are described in the Mesh Architecture Guide [draft-hallambaker-mesh-architecture] .

2.3. Related Specifications

The architecture of the Mathematical Mesh is described in the Mesh Architecture Guide [draft-hallambaker-mesh-architecture] . The Mesh documentation set and related specifications are described in this document.

2.4. Implementation Status

The implementation status of the reference code base is described in the companion document [draft-hallambaker-mesh-developer] .

3. Mesh Assertions

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

Mesh Assertions are divided into two classes. Mesh Profiles are self-signed assertions. Assertions that are not self-signed are called declarations. The only type of declaration currently defined is a Connection Declaration describing the connection of one profile to another. Currently, five profile and four connection types are defined:

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html [2].]]

Profiles And Connections

3.1. Encoding

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

Identifier An identifier for the assertion.

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

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

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

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

Note that the implementation of Conditions differs significantly from that of SAML. Relying parties are required to process condition clauses in a SAML assertion to determine validity. Mesh Relying parties MAY verify the conditions clauses or rely on the trustworthiness of the provider.

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

3.2. Mesh Profiles

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

- Profiles describe credentials, they do not make identity statements

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

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

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

It should be noted however that the role of Certificate Authorities is redefined rather than eliminated. Rather than making assertions whose subject is represented by identities which are inherently

mutable and subjective, Certificate Authorities can now make assertions about immutable cryptographic keys.

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

A Profile is valid if and only if:

- There is a SignatureKey field.

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

A profile has the status current if and only if:

- The Profile is valid

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

3.3. Mesh Connections

3.4. Mesh Private Declarations

4. Architecture

The Mesh architecture has four principal components:

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

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

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

Mesh Messaging

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

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

The separation of accounts and services as separate components is a key distinction between the Mesh and earlier Internet applications. A Mesh account belongs to the owner of the Mesh and not the account service provider which the user may change at any time of their choosing. A Mesh account may be connected to multiple service providers to provide backup capability or to none.

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

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html [3].]]

Alice's Personal Mesh

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

4.1. Device Management

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

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

4.1.1. Master Profile

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

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

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html [4].]]

Master Profile and Associated Device and Account Connection Assertions.

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

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

```
{
  "ProfileMaster":{
    "KeySignature":{
      "UDF":"MBYS-BF7J-OCV2-42M6-BWTH-2QUK-FZG3",
      "PublicParameters":{
        "PublicKeyECDH":{
          "crv":"Ed448",
          "Public":"eBYEyA6nN0BQKFJcY7kg8f4_kVGP2GmC1F_tzhLwe0oH1so
  qG8OEtKrm5rkyvs87aBTXC4Fjt3wA"}}},
    "OnlineSignatureKeys":[{
        "UDF":"MDJS-HG6T-W3TE-UZKO-LDBY-5TGM-TAGT",
        "PublicParameters":{
          "PublicKeyECDH":{
            "crv":"Ed448",
            "Public":"4zqbLuaOBu6Y7ywkuLzGiEW3NVRo65vX6KXUaogizo9ob
  DLWFdLjqLm1vozc6BYCkfbPCOq1VniA"}}}
      ],
    "KeyEncryption":{
      "UDF":"MDAK-ZLB4-ESKG-IDKP-OSKE-OHOS-IOFP",
      "PublicParameters":{
        "PublicKeyECDH":{
          "crv":"Ed448",
          "Public":"rOwpwjCoC20VCiaeYIJdMG5HehxtZ4LRIjexuLE04j4DVZG
  y-zBmsRlQt6ra4ml1-66qVfzFt9QA"}}}}}
```

4.1.1.1. Creating a ProfileMaster

Creating a ProfileMaster comprises the steps of:

1. Creating a Master Signature key.

2. Creating an Online Signing Key

3. Signing the ProfileMaster using the Master Signature Key

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

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

4.1.1.2. Updating a ProfileMaster

Updating a ProfileMaster comprises the steps of:

1. Making the necessary changes.

2. Signing the ProfileMaster using the Master Signature Key

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

4.1.1.3. The Device Catalog

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

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

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

4.1.2. Mesh Devices

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

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

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

A device profile provides the axiom of trust and the key contributions of the device:

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html [5].]]

Mapping of Device Profile and Device Private to Device Connection Keys.

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

Alice's Device Profile specifies keys for encryption, signature and exchange:

Since the Device Profile keys are ultimately under the control of the device and/or software provider, these are considered insufficiently trustworthy and the administration device creates key contributions to be added to the device keys to establish the key set to be used in the context of the user's personal Mesh:

$$$$ Empty $$$$

The resulting key set is specified in the device connection:

$$$$ Empty $$$$

All the above are combined to form the CatalogEntryDevice entry:

```
{
```

```
"CatalogEntryDevice":{
  "UDF":"MBRF-BFUO-R765-3KQP-DXNK-TOGV-65YA",
  "DeviceUDF":"MDBS-UDDJ-FKMR-FY3G-NT75-KREE-7GOM",
  "EnvelopedProfileDevice":[{
      "dig":"S512",
      "cty":"application/mmm"},
    "ewogICJQcm9maWxlRGV2aWNlIjogewogICAgIktleVNpZ25hdHVyZSI6IHsK
ICAgICAgIlVERiI6ICJNREJTLVVEREotRktNUi1GWTNHLU5UNzUtS1JFRS03R09NI
iwKICAgICAgIlB1YmxpY1BhcmFtZXRlcnMiOiB7CiAgICAgICAgIlB1YmxpY0tleU
VDREgiOiB7CiAgICAgICAgICAiY3J2IjogIkVkNDQ4IiwKICAgICAgICAgICJQdWJ
saWMiOiAiZnFOQWFLMEJ1WU9PeDhkV0dzc1gyWkZCeGF1bjlZZlMtNS1QTWR0b3c0
emJTLS0weGJmdQogIGxkbGxBTlVFQkl1ajJoWjY3RUl4WlZNQSJ9fX0sCiAgICAiS
2V5RW5jcnlwdGlvbiI6IHsKICAgICAgIlVERiI6ICJNRFJNLVdCUk4tNDJQUS1WRD
RGLVlWSDItRklaNy1HRkVHIiwKICAgICAgIlB1YmxpY1BhcmFtZXRlcnMiOiB7CiA
gICAgICAgIlB1YmxpY0tleUVDREgiOiB7CiAgICAgICAgICAiY3J2IjogIkVkNDQ4
IiwKICAgICAgICAgICJQdWJsaWMiOiAiRldJckRPNUdnRVVwTnE3YWpjTEx0VVk5V
XloMC1qaTF5a0RYcHZxZjhXenA2dHo3UzQxZgogIHl4Nzc3X0pyanpzTURKUGZfWE
Z6TlphQSJ9fX0sCiAgICAiS2V5QXV0aGVudGljYXRpb24iOiB7CiAgICAgICJVREY
iOiAiTUQ3NC1RUVBBLUlOQVUtWVRKTi1FTFlVLVZQN0QtTlpRRCIsCiAgICAgICJQ
dWJsaWNQYXJhbWV0ZXJzIjogewogICAgICAgICJQdWJsaWNLZXlFQ0RIIjogewogI
CAgICAgICAgImNydiI6ICJFZDQ0OCIsCiAgICAgICAgICAiUHVibGljIjogImd0X3
gzbERmR0ppaTFCRVVmU09KRlQwSFNob1U4T1hzZFd6Sm9XT3UyRTRiUTFiWWxpZks
KICBsR1VWdTNsX2hzV0p6UTdsRUNZNG9ZcUEifX19fX0",
    {
      "signatures":[{
          "signature":"U6l-ZRptqcZuU3f5YnQ3ziYIki3R6M1SSTxE8Ybpni
gTkTV7wUz8OZtEtsuAKTch9U8yh74ymXGAN40GofhMoOVocwcB-mcrcmsgQt5IEvH
7TS7G4bLCbQA8_9irpYW7XMXh8c94RaayLwH6yRn43ysA"}
        ],
      "PayloadDigest":"otiC41Z57sm74AZt4z81_09sgWbigOXzXBJsJbom0a
Dx9TK02RTfkQ_6s0paCD3-yt1cI2uYt499mSJwh9CZ-g"}
    ],
  "EnvelopedDeviceConnection":[{
      "dig":"S512"},
    "ewogICJBc3NlcnRpb25EZXZpY2VDb25uZWN0aW9uIjogewogICAgIktleVNp
Z25hdHVyZSI6IHsKICAgICAgIlVERiI6ICJNQlJGLUJGVU8tUjc2NS0zS1FQLURYT
kstVE9HVi02NVlBIiwKICAgICAgIlB1YmxpY1BhcmFtZXRlcnMiOiB7CiAgICAgIC
AgIlB1YmxpY0tleUVDREgiOiB7CiAgICAgICAgICAiY3J2IjogIkVkNDQ4IiwKICA
gICAgICAgICJQdWJsaWMiOiAiNEhQRHJGSTBmNjJlOElpSzJzSVhDQkdzaWljMHow
Y2IyVER2MThwZFRKOG4xX1JEeHhKaQogIG5HUDR0YkxCRkc4ejVfQXp3dHJzcG9hQ
SJ9fX0sCiAgICAiS2V5RW5jcnlwdGlvbiI6IHsKICAgICAgIlVERiI6ICJNRFkyLV
NWNkEtRFgzNi03WjJELVJNTEEtMkdZRC1RU0UyIiwKICAgICAgIlB1YmxpY1BhcmF
tZXRlcnMiOiB7CiAgICAgICAgIlB1YmxpY0tleUVDREgiOiB7CiAgICAgICAgICAi
Y3J2IjogIkVkNDQ4IiwKICAgICAgICAgICJQdWJsaWMiOiAiZGRvN1FGdGRWTndiX
2d0T1Q4cjQ0YUhvbUp4UFowY2k2aHZrTlU0WjVnNmRJMHBaNGkxQQogIGJwZ2FHT0
txRWJkWE5Gcy1zNlNGUkNDQSJ9fX0sCiAgICAiS2V5QXV0aGVudGljYXRpb24iOiB
7CiAgICAgICJVREYiOiAiTUNJRy1YT1E2LUFUNkctU1FYQS1QQlZFLUgyM0ctNFNF
WSIsCiAgICAgICJQdWJsaWNQYXJhbWV0ZXJzIjogewogICAgICAgICJQdWJsaWNLZ
```

```
XlFQ0RIIjogewogICAgICAgICAgImNydiI6ICJFZDQ0OCIsCiAgICAgICAgICAiUH
VibGljIjogIndyUkxuaDdCOThiZDdCWm9wSU1CODNsSE5iR2tCVUJLelhaanIwMmh
CV05jVFg3UTYtLWsKICBSNGNsSV9iVU1HbUVaazllbkpMN2d2S0EifX19fX0",
    {
      "signatures":[{
          "signature":"_iPhuGpv8Xh039rEoXIuyVxy_A8ZwYZpd6CerkRZKD
vMb7MB4rofIuDlBvqyPkxmhmDHhYojNJMAUTUAPGtWLkDh8RVIXCatF9bGFsai1pc
0CeppVc6rMRirGu4SkaGrntQ7pAflO1xCxb-71ezguAcA"}
        ],
      "PayloadDigest":"BNkv4-z9EagIVTM0ogNIIPbB7_9PPObDt7CQEwv0yp
zf_I2aUHffg12gH4owdH7xcVh7AHvKI-FVCwvxElHVag"}
    ],
  "EnvelopedDevicePrivate":[{
      "enc":"none",
      "Salt":"jUuy9EkwLkMF-l0KDVG-QA",
      "cty":"application/mmm",
      "recipients":[{
          "kid":"MDRM-WBRN-42PQ-VD4F-YVH2-FIZ7-GFEG",
          "epk":{
            "PublicKeyECDH":{
              "crv":"Ed448",
              "Public":"Uf3BoP6qJayi3nNelfefiQ5R_YP1boOdq-Gj86Skk
hmTpyCiDkI8lLNIWDrFrxdahMhuYyYEComA"}},
          "wmk":"pqampqampqY"},
        {
          "kid":"MDAK-ZLB4-ESKG-IDKP-OSKE-OHOS-IOFP",
          "epk":{
            "PublicKeyECDH":{
              "crv":"Ed448",
              "Public":"jKmoG_Z8Tlr2IkhA_65dgAAhPCk1MtS9mfRk97Ux8
5AbfwKn4Gi6eJxphdPGRbzsGc74J5VB6SSA"}},
          "wmk":"pqampqampqY"}
        ]},
    "ewogICJBc3NlcnRpb25EZXZpY2VQcml2YXRlIjogewogICAgIktleVNpZ25h
dHVyZSI6IHsKICAgICAgIlVERiI6ICJNQlJGLUJGVU8tUjc2NS0zS1FQLURYTkstV
E9HVi02NVlBIiwKICAgICAgIkJhc2VVREYiOiAiTURCUy1VRERKLUZLTVItRlkzRy
1OVDc1LUtSRUUtN0dPTSIsCiAgICAgICJPdmVybGF5IjogewogICAgICAgICJQcml
2YXRlS2V5RUNESCI6IHsKICAgICAgICAgICJjcnYiOiAiRWQ0NDgiLAogICAgICAg
ICAgIlByaXZhdGUiOiAiOE1nZUducXBrVjdMd3NNcW5LbHQtY24zTktFd2ZBeHpBM
TVGR2cxWmVsSjJfdnhTRVNmCiAgQmRfa2g0OHFPeEpUMFNxQnNMd01tU0tZIn19fS
wKICAgICJLZXlFbmNyeXB0aW9uIjogewogICAgICAiVURGIjogIk1EWTItU1Y2QS1
EWDM2LTdaMkQtUk1MQS0yR1lELVFTRTIiLAogICAgICAiQmFzZVVERiI6ICJNRFJN
LVdCUk4tNDJQUS1WRDRGLVlWSDItRklaNy1HRkVHIiwKICAgICAgIk92ZXJsYXkiO
iB7CiAgICAgICAgIlByaXZhdGVLZXlFQ0RIIjogewogICAgICAgICAgImNydiI6IC
JFZDQ0OCIsCiAgICAgICAgICAiUHJpdmF0ZSI6ICJrOTR1UUQxSXUwcnd6UkNrU3h
3cWJHNDU3R1Rka1pmSF9HQXJqZS1UUmZLZTdkd0oyWTIKICBBYU9lb1g5X2VpWE5D
UEZIQ2xvS1ZFUE0ifX19LAogICAgIktleUF1dGhlbnRpY2F0aW9uIjogewogICAgI
CAiVURGIjogIk1DSUctWE9RNi1BVDZHLVNRWEEtUEJWRS1IMjNHLTRTRVkiLAogIC
```

```
     AgICAiQmFzZVVERiI6ICJNRDc0LVFRUEEtSU5BVS1ZVEpOLUVMWVUtVlA3RC1OWlF
     EIiwKICAgICAgIk92ZXJsYXkiOiB7CiAgICAgICAgIlByaXZhdGVLZXlFQ0RIIjog
     ewogICAgICAgICAgImNydiI6ICJFZDQ0OCIsCiAgICAgICAgICAiUHJpdmF0ZSI6I
     CJkUmhudWJOa1o3ekFMdWlBR2FWWExlWE1JNVRFNDMwejZWWk1sTjRnSTBDZFVLSk
     0yLUMKICBHU3VMTGVQQUltOUw5TktneldaTUNEVU0ifX19fX0"
         ]}}
```

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

4.1.2.1. Creating a ProfileDevice

Creating a ProfileDevice comprises the steps of:

1. Creating the necessary key

2. Signing the ProfileDevice using the Master Signature Key

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

4.1.2.2. Connection to a Personal Mesh

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

1. Generating the PrivateDevice keys.

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

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

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

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

4.2. Mesh Accounts

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

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

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

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

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html [6].]]

Account Profile Connected to Devices and Services.

$$$$ Empty $$$$

4.2.1. Creating a ProfileAccount

Creating a ProfileAccount comprises the steps of:

1. [TBS]

2. .

3. Signing the ProfileMaster using the Master Signature Key

4.2.2. Connecting a Device to an Account

Adding a device to an account comprises the steps of:

1. Creating a PrivateAccount instance for the device.

2. Creating a ConnectionAccountDevice for the device using the public keys from the PrivateAccount instance and the ProfileDevice.

3. Creating an ActivationAccount for the device containing the PrivateAccount and ConnectionAccountDevice instances.

4. Adding the ActivationAccount to the CatalogEntryDevice of the device.

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

4.2.3. Binding and Account to a Service

Binding a ProfileAccount to a Mesh Service the steps of:

1. [TBS]

2. .

3. Signing the ProfileMaster using the Master Signature Key

4.3. Mesh Services

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

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html [7].]]

Service Profile and Delegated Host Assertion.

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

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

$$$$ Empty $$$$

4.3.1. Creating a ProfileService

[TBS]

Creating a ProfileService comprises the steps of:

1. [TBS]

2. .

3. [TBS]

4.

5. Signing the ProfileMaster using the Master Signature Key

4.3.2. Creating a ProfileHost

Creating a ProfileHost comprises the steps of:

1. [TBS]

2. .

3. [TBS]

4.

5. Signing the ConnectionHost using the Master Signature Key of the ProfileService.

4.3.3. Creating a ConnectionHost

Creating a ConnectionHost comprises the steps of:

1. [TBS]

2. .

3. Signing the ConnectionHost using the Master Signature Key of the ProfileService.

4.4. Mesh Messaging

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

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

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html [8].]]

Performing Access Control on Outbound Messages

The outbound Mesh Service checks to see that the message request does not violate its acceptable use policy. Accounts that make a large number of message requests that result in complaints SHOULD be subject to consequences ranging from restriction of the number and type of messages sent to suspending or terminating messaging privileges.

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html [9].]]

Performing Access Control on Outbound Messages

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

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

[[This figure is not viewable in this format. The figure is available at http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html [10].]]

Performing Access Control on Inbound Messages

4.4.1. Traffic Analysis

The Mesh Messaging protocol as currently specified provides only limited protection against traffic analysis attacks. The use of TLS to encrypt communication between Mesh Services limits the effectiveness of na?ve traffic analysis mechanisms but does not prevent timing attacks unless dummy traffic is introduced to obfuscate traffic flows.

The limitation of the message size is in part intended to facilitate use of mechanisms capable of providing high levels of traffic analysis such as mixmaster and onion routing but the current Mesh

Service Protocol does not provide support for such approaches and there are no immediate plans to do so.

5. Mesh Catalogs

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

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

<<prefix>_<name>

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

The following catalogs are currently specified within the Mathematical Mesh.

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

Device: mmm_CatalogDevice Contains descriptions of devices connected to the account and the permissions assigned to them

Contact: mmm_CatalogContact Contains logical and physical contact information for people and organizations.

Credential: mmm_CatalogCredential Contains credentials used to access network resources.

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

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

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

In many cases, the Mesh Catalog offers capabilities that represent a superset of the capabilities of an existing application. For example, the task catalog supports the appointment tracking functions

of a traditional calendar application and the task tracking function of the traditional 'to do list' application. Combining these functions allows tasks to be triggered by other events other than the passage of time such as completion of other tasks, geographical presence, etc.

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

5.1. Application

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

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

5.1.1. Mesh Account

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

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

5.1.2. SSH

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

A user may have separate SSH configurations for separate purposes within a single Mesh Account. This allows a system administrator servicing multiple clients to maintain separate SSH profiles for each

of her customers allowing credentials to be easily (and verifiably) revoked at contract termination.

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

$$$$ Empty $$$$

5.1.3. Mail

Mail configuration profiles are described by one or more CatalogEntryApplicationMail entries, one for each email account connected to the Mesh profile. The corresponding activation records for the connected devices contain information used to provide the device with the necessary decryption information.

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

$$$$ Empty $$$$

5.2. Device

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

The management of the device catalog is described in the section describing Mesh Device Management.

5.3. Contact

The contacts catalog contains CatalogEntryContact entries which describe

$$$$ Empty $$$$

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

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

5.4. Credential

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

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

$$$$ Empty $$$$

5.5. Bookmark

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

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

$$$$ Empty $$$$

5.6. Task

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

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

$$$$ Empty $$$$

5.7. Network

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

$$$$ Empty $$$$

6. Mesh Messages

All communications between Mesh accounts takes the form of a Mesh Message carried in a Dare Envelope. Mesh Messages are stored in two spools associated with the account, the SpoolOutbound and the SpoolInbound containing the messages sent and received respectively.

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

6.1. Completion

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

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

[NYI]

6.2. Connection

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

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

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

Further details of this mechanism are described in [draft-hallambaker-mesh-protocol] .

[NYI]

6.3. Contact

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

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

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

[NYI]

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

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

6.4. Confirmation

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

Two confirmation messages are specified, a request and response.

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

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

[NYI]

7. Schema

7.1. Shared Classes

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

7.1.1. Structure: PublicKey

The PublicKey class is used to describe public key pairs and trust assertions associated with a public key.

UDF: String (Optional) UDF fingerprint of the public key parameters/

X509Certificate: Binary (Optional) List of X.509 Certificates

X509Chain: Binary [0..Many] X.509 Certificate chain.

X509CSR: Binary (Optional) X.509 Certificate Signing Request.

7.2. Mesh Assertion Objects

Base class for all Mesh Assertion objects.

7.2.1. Structure: Assertion

Parent class from which all assertion classes are derived

Names: String [0..Many] Fingerprints of index terms for profile retrieval. The use of the fingerprint of the name rather than the name itself is a precaution against enumeration attacks and other forms of abuse.

Updated: DateTime (Optional) The time instant the profile was last modified.

NotaryToken: String (Optional) A Uniform Notary Token providing evidence that a signature was performed after the notary token was created.

7.2.2. Structure: Condition

Parent class from which all condition classes are derived.

[No fields]

7.2.3. Structure: Profile

Inherits: Assertion

Parent class from which all profile classes are derived

SignatureKey: PublicKey (Optional) The signature key associated with the profile.

7.2.4. Keyset Classes

7.2.5. Structure: EscrowedKeySet

A set of escrowed keys.

[No fields]

7.2.6. Profile Classes

7.2.7. Structure: ProfileMaster

Inherits: Profile

Describes the long term parameters associated with a personal profile.

This profile MUST be signed by

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 OSK which is used to sign device administration application profiles.

7.2.8. Structure: ProfileDevice

Inherits: Profile

Describes a mesh device.

This profile MUST be signed by the DeviceSignatureKey

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

7.2.9. Structure: ProfileApplication

Inherits: Profile

Contains the public description of a Mesh application.

[No fields]

7.2.10. Structure: ProfileMesh

Inherits: ProfileApplication

Contains the binding of a device to a MasterProfile. Each device has a separate profile which MUST be signed by an OnlineSignatureKey

Account: String (Optional) Account address.

MasterProfile: DareEnvelope (Optional) Master profile of the account being registered.

AccountEncryptionKey: PublicKey (Optional) Key used to encrypt data under this profile

7.2.11. Structure: ProfileMeshDevicePublic

Inherits: ProfileApplication

Inherits: ProfileApplication

DeviceProfile: DareEnvelope (Optional) Device profile of the device making the request.

Permissions: Permission [0..Many] List of the permissions that the device has been granted.

7.2.12. Structure: ProfileMeshDevicePrivate

Inherits: ProfileApplication

Inherits: ProfileApplication

Permissions: Permission [0..Many] List of the permissions that the device has been granted.

ProfileNonce: Binary (Optional) Random nonce used to mask the fingerprint of the profile UDF.

ProfileWitness: Binary (Optional) Witness value calculated over the ProfileNonce and profile UDF

7.2.13. Structure: DeviceRecryptionKey

UDF: String (Optional) The fingerprint of the encryption key

RecryptionKey: PublicKey (Optional) The recryption key

DeviceRecryptionKeyEncrypted: DareEnvelope (Optional) The decryption key encrypted under the user's device key.

7.3. Common Structures

7.3.1. Structure: Permission

Name: String (Optional)

Name: String (Optional)

Role: String (Optional)

Role: String (Optional)

Capabilities: DareEnvelope (Optional) Keys or key contributions enabling the operation to be performed

7.3.2. Structure: Contact

Identifier: String (Optional)

Identifier: String (Optional)

Account: String (Optional)

Account: String (Optional)

FullName: String (Optional)

FullName: String (Optional)

Title: String (Optional)

Title: String (Optional)

First: String (Optional)

First: String (Optional)

Middle: String (Optional)

Middle: String (Optional)

Last: String (Optional)

Last: String (Optional)

Suffix: String (Optional)

Suffix: String (Optional)

Labels: String [0..Many]

Labels: String [0..Many]

Addresses: Address [0..Many]

Addresses: Address [0..Many]

Locations: Location [0..Many]

Locations: Location [0..Many]

Roles: Role [0..Many]

7.3.3. Structure: Role

CompanyName: String (Optional)

CompanyName: String (Optional)

Addresses: Address [0..Many]

Addresses: Address [0..Many]

Locations: Location [0..Many]

7.3.4. Structure: Address

URI: String (Optional)

URI: String (Optional)

Labels: String [0..Many]

7.3.5. Structure: Location

Appartment: String (Optional)

Appartment: String (Optional)

Street: String (Optional)

Street: String (Optional)

District: String (Optional)

District: String (Optional)

Locality: String (Optional)

Locality: String (Optional)

County: String (Optional)

County: String (Optional)

Postcode: String (Optional)

Postcode: String (Optional)

Country: String (Optional)

7.3.6. Structure: Reference

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

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

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

7.4. Catalog Entries

7.4.1. Structure: CatalogEntry

[No fields]

7.4.2. Structure: CatalogEntryDevice

Inherits: CatalogEntry

Public device entry, indexed under the device ID

Account: String (Optional) The Account to which this entry binds this device.

UDF: String (Optional) UDF of the signature key

AuthUDF: String (Optional) UDF of the authentication ID

ProfileMeshDevicePublicSigned: DareEnvelope (Optional) The device profile

ProfileMeshDevicePrivateEncrypted: DareEnvelope (Optional) The device profile

DeviceRecryptionKeys: DeviceRecryptionKey [0..Many] Decryption key entries.

7.4.3. Structure: CatalogEntryCredential

Inherits: CatalogEntry

Inherits: CatalogEntry

Protocol: String (Optional)

Protocol: String (Optional)

Service: String (Optional)

Service: String (Optional)

Username: String (Optional)

Username: String (Optional)

Password: String (Optional)

7.4.4. Structure: CatalogEntryNetwork

Inherits: CatalogEntry

Inherits: CatalogEntry

Protocol: String (Optional)

Protocol: String (Optional)

Service: String (Optional)

Service: String (Optional)

Username: String (Optional)

Username: String (Optional)

Password: String (Optional)

7.4.5. Structure: CatalogEntryContact

Inherits: CatalogEntry

Inherits: CatalogEntry

Key: String (Optional) Unique key.

Permissions: Permission [0..Many] List of the permissions that the contact has been granted.

Contact: DareEnvelope (Optional) The (signed) contact data.

7.4.6. Structure: CatalogEntryContactRecryption

Inherits: CatalogEntryContact

[No fields]

7.4.7. Structure: CatalogEntryBookmark

Inherits: CatalogEntry

Inherits: CatalogEntry

Uri: String (Optional)

Uri: String (Optional)

Title: String (Optional)

Title: String (Optional)

Path: String (Optional)

7.4.8. Structure: CatalogEntryTask

Inherits: CatalogEntry

Inherits: CatalogEntry

Task: DareEnvelope (Optional)

Task: DareEnvelope (Optional)

Key: String (Optional) Unique key.

7.4.9. Structure: Task

Key: String (Optional) Unique key.

Start: DateTime (Optional)

Start: DateTime (Optional)

Finish: DateTime (Optional)

Finish: DateTime (Optional)

StartTravel: String (Optional)

StartTravel: String (Optional)

FinishTravel: String (Optional)

FinishTravel: String (Optional)

TimeZone: String (Optional)

TimeZone: String (Optional)

Title: String (Optional)

Title: String (Optional)

Description: String (Optional)

Description: String (Optional)

Location: String (Optional)

Location: String (Optional)

Trigger: String [0..Many]

Trigger: String [0..Many]

Conference: String [0..Many]

Conference: String [0..Many]

Repeat: String (Optional)

Repeat: String (Optional)

Busy: Boolean (Optional)

7.4.10. Structure: CatalogEntryApplication

Inherits: CatalogEntry

Inherits: CatalogEntry

Key: String (Optional)

7.4.11. Structure: CatalogEntryApplicationEntry

[No fields]

7.4.12. Structure: CatalogEntryApplicationRecryption

[No fields]

7.4.13. Structure: CatalogEntryApplicationSSH

[No fields]

7.4.14. Structure: CatalogEntryApplicationMail

[No fields]

7.4.15. Structure: CatalogEntryApplicationNetwork

[No fields]

7.5. Messages

7.5.1. Structure: MeshMessage

MessageID: String (Optional)

MessageID: String (Optional)

Sender: String (Optional)

Sender: String (Optional)

Recipient: String (Optional)

Recipient: String (Optional)

References: Reference [0..Many]

7.5.2. Structure: MeshMessageComplete

Inherits: MeshMessage

[No fields]

7.5.3. Structure: MessageConnectionRequest

Inherits: MeshMessage

Inherits: MeshMessage

Account: String (Optional)

Account: String (Optional)

DeviceProfile: DareEnvelope (Optional) Device profile of the device making the request.

ClientNonce: Binary (Optional)

ClientNonce: Binary (Optional)

ServerNonce: Binary (Optional)

ServerNonce: Binary (Optional)

Witness: String (Optional)

Witness: String (Optional)

PinID: String (Optional) Pin identifier used to identify a PIN authenticated request.

7.5.4. Structure: MessageConnectionPIN

Inherits: MeshMessage

Inherits: MeshMessage

Account: String (Optional)

Account: String (Optional)

Expires: DateTime (Optional)

Expires: DateTime (Optional)

PIN: String (Optional)

7.5.5. Structure: MessageContactRequest

Inherits: MeshMessage

Inherits: MeshMessage

Contact: DareEnvelope (Optional) The contact data.

7.5.6. Structure: MessageConfirmationRequest

Inherits: MeshMessage

Inherits: MeshMessage

Text: String (Optional)

7.5.7. Structure: MessageConfirmationResponse

Inherits: MeshMessage

Inherits: MeshMessage

ResponseID: String (Optional)

ResponseID: String (Optional)

Accept: Boolean (Optional)

7.5.8. Structure: MessageTaskRequest

Inherits: MeshMessage

[No fields]

8. Security Considerations

The security considerations for use and implementation of Mesh services and applications are described in the Mesh Security Considerations guide [draft-hallambaker-mesh-security] .

9. IANA Considerations

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

10. Acknowledgements

A list of people who have contributed to the design of the Mesh is presented in [draft-hallambaker-mesh-architecture] .

11. References

11.1. Normative References

[draft-hallambaker-mesh-architecture]
Hallam-Baker, P., "Mathematical Mesh Part I: Architecture Guide", draft-hallambaker-mesh-architecture-07 (work in progress), April 2019.

[draft-hallambaker-mesh-cryptography]
Hallam-Baker, P., "Mathematical Mesh Part VIII: Cryptographic Algorithms", draft-hallambaker-mesh-cryptography-00 (work in progress), April 2019.

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

[draft-hallambaker-mesh-protocol]
Hallam-Baker, P., "Mathematical Mesh Part V: Protocol Reference", draft-hallambaker-mesh-protocol-00 (work in progress), April 2019.

[draft-hallambaker-mesh-security]
Hallam-Baker, P., "Mathematical Mesh Part VII: Security Considerations", draft-hallambaker-mesh-security-00 (work in progress), April 2019.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997.

11.2. Informative References

[draft-hallambaker-mesh-developer]
Hallam-Baker, P., "Mathematical Mesh: Reference Implementation", draft-hallambaker-mesh-developer-08 (work in progress), April 2019.

[draft-hallambaker-mesh-trust]
Hallam-Baker, P., "Mathematical Mesh Part VI: The Trust Mesh", draft-hallambaker-mesh-trust-01 (work in progress), April 2019.

[RFC2426] Dawson, F. and T. Howes, "vCard MIME Directory Profile", RFC 2426, DOI 10.17487/RFC2426, September 1998.

[RFC5545] Desruisseaux, B., "Internet Calendaring and Scheduling Core Object Specification (iCalendar)", RFC 5545, DOI 10.17487/RFC5545, September 2009.

11.3. URIs

[1] http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html

[2] http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html

[3] http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html

[4] http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html

[5] http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html

[6] http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html

[7] http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html

[8] http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html

[9] http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html

[10] http://mathmesh.com/Documents/draft-hallambaker-mesh-schema.html

Author's Address

Phillip Hallam-Baker

Email: phill@hallambaker.com