Workgroup: Network Working Group
Internet-Draft:
draft-hallambaker-mesh-developer
Published: 28 June 2023
Intended Status: Informational
Expires: 30 December 2023
Authors: P. M. Hallam-Baker

Mathematical Mesh: Reference Implementation

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.

This document describes the Mesh reference code and how to install, run and make use of it in applications. It does not form a part of the Mesh specifications and is not normative.

This document is also available online at http://mathmesh.com/Documents/draft-hallambaker-mesh-developer.html.

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 https://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 30 December 2023.

carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Definitions
  1.1. Requirements Language
  1.2. Defined Terms
  1.3. Related Specifications
  1.4. Implementation Status
2. Getting the Reference Code and Build Tools
  2.1. Obtaining the Development Environment
  2.2. Obtaining the Build Tools
  2.3. Obtaining the Mesh Source Libraries
3. Compiling the Reference Code
  3.1. Creating a software signing key
  3.2. Create (dummy) build action files
4. Running the Reference Code Examples
  4.1. Starting the Server
  4.2. The Profile Manager Wizard
  4.3. The Profile Connection Wizard
5. Platform specific configuration data
  5.1. Windows
    5.1.1. Private Key Data
    5.1.2. Registry settings
    5.1.3. Profile data files
  5.2. OSX and Linux
6. Implementation Status
  6.1. Reference Implementation
    6.1.1. Coverage:
    6.1.2. Licensing
    6.1.3. Implementation Experience
    6.1.4. Contact Info
7. Security Considerations
8. IANA Considerations
9. Acknowledgements
10. Informative References

1. 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.

1.1. Requirements Language

This document is not normative and does not contain requirements language

1.2. Defined Terms

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

1.3. 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.

1.4. Implementation Status

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

2. Getting the Reference Code and Build Tools

The Mesh Reference library was developed using Visual Studio 2017 Community Edition [VS2017] using PHB's Build Tools [PHB2017] extensions. The reference code itself is currently limited to C# libraries.

The code should in theory run under other operating systems but this has not been tested recently.

Development under different development environments is also possible but would require re-engineering to make use of the line mode versions of the build tools.

2.1. Obtaining the Development Environment

Visual Studio 2017 Community Edition is currently available at no cost for a wide range of non-commercial development including personal use and development of Open Source software. For full details, please consult the license published by Microsoft.

https://www.visualstudio.com/

2.2. Obtaining the Build Tools

Over half the code in the reference code library is generated using code generators. These are used to ensure that the specification, examples and reference code are always kept in synchronization.

The build tools are published under an MIT License and are available in two forms:

As stand-alone tools to be run from the command line.

As a VSIX package that integrates into the Visual Studio environment.

The source distribution is configured to use the tools integrated into the Visual Studio environment. If development on other platforms is desired, the simplest approach is likely to be to write a tool that reads the Visual Studio configuration files and generates the corresponding files for use with make.

The VSIX package is available from the Visual Studio extensions gallery:

PHB Code Generation Tools

The source code for the build tools is available from:

https://sourceforge.net/projects/phb-build-tools/

2.3. Obtaining the Mesh Source Libraries

The Mesh reference library source code is published under an MIT license and is available from:

https://sourceforge.net/projects/mathematicalmesh/

3. Compiling the Reference Code

To compile the code it is necessary to

Create a signing key

Create batch files for pre and post build tasks

3.1. Creating a software signing key

The purpose of signing assemblies is so that they can be authenticated during the load process. For this to be secure, it is of course essential that each developer has their own key.

To create a software developer signing key, the Visual Studio 'sn' tool is used. To run the tool, start the Visual Studio Developer Console *in administrator mode*. This requires the following steps:

Move to a directory you want to write to.

Set the machine to create user based keys

Create a new key and write it to a file.

Copy the file from the key to a container.

Delete the container.

Locate the private key file

Give permission to use the key.

This is of course one of the tasks we would like to automate with the Mesh tools in due course but that presents a bootstrap problem.

```
C:\Windows\System32>cd hallam

c:\Users\hallam>sn -m N

Microsoft (R) .NET Framework Strong Name Utility  Version 4.0.30319.0
Copyright (c) Microsoft Corporation.  All rights reserved.

Key containers are user based

c:\Users\hallam>sn -k fred.snk

Microsoft (R) .NET Framework Strong Name Utility  Version 4.0.30319.0
Copyright (c) Microsoft Corporation.  All rights reserved.

Key pair written to fred.snk

c:\Users\hallam>sn -i fred.snk SigningKeyDeveloper

Microsoft (R) .NET Framework Strong Name Utility  Version 4.0.30319.0
Copyright (c) Microsoft Corporation.  All rights reserved.

Key pair installed into 'SigningKeyDeveloper'

c:\Users\hallam>del fred.snk

c:\Users\hallam>
```

3.2. Create (dummy) build action files

Visual Studio allows projects to specify batch files to be run before and after a project build. Since the actions to be taken are likely to change from developer to developer, these are specified in separate batch files. All that is necessary to build the code without warnings is to specify a set of dummy batch files with the following names and place them somewhere in the command line $PATH environment variable.

The files required are:

VSPreBuild.bat

VSPostBuild.bat

VSPostBuildWindows.bat

VSPostBuildOSX.bat

VSPostBuildLinux.bat

The following code will prevent error messages being thrown:

```
@echo off
SETLOCAL
exit /b 0
```

4. Running the Reference Code Examples

The reference code examples are designed to illustrate how the Mesh might be used in an application rather than be standalone tools in their own right. The Mesh is designed to make it each for developers to add security to their own applications rather than providing the applications themselves.

4.1. Starting the Server

On the Windows platform, the server runs in the context of the platform Web server and must be granted permission to bind to the range of server addresses used using the netsh command.

From a command prompt with administrator privileges, run the following command:

```
netsh http add urlacl http://<domain>/.well-known/mmm/
    \user=<machine>\<user>
```

Where <domain> is the DNS domain name under which the service is run, <machine> is the Windows domain name of the machine and <user> the account name.

To start the service from the command line type:

```
servermesh <domain>
```

The server does not require administration privileges.

4.2. The Profile Manager Wizard

The profile manager wizard demonstrates functions that are performed on an administration device. These include creating a completely new profile and initial configuration of applications, connecting a device to the profile and recovery of the profile from escrow data.

To run the client from the command line, place the executable image in a location that it will be found in the PATH variable and type:

```
meshclient
```

4.3. The Profile Connection Wizard

The Profile connection wizard demonstrates the much more restricted functionality that would be required in a Mesh connected application and/or a profile manager for a non-administration device.

To run the client from the command line, place the executable image in a location that it will be found in the PATH variable and type:

```
meshconnect
```

5. Platform specific configuration data

5.1. Windows

5.1.1. Private Key Data

All private key data is stored using the Windows public key store. At minimum, this ensures that private keys are obfuscated and encrypted under the account password to protect the data against casual extraction attacks. On a machine with cryptographic hardware support such as a TPM or HSM, extraction of the private key may be infeasible without physical access to the machine and possibly require sophisticated diagnostic equipment.

5.1.2. Registry settings

Separate settings are used for production and test code. Test Code should use the Registry Hive:

HKEY_CURRENT_USER\SOFTWARE\CryptoMesh

Production code should use the hive

HKEY_CURRENT_USER\SOFTWARE\MathematicalMesh

In either case the sub structure is:

Accounts Contains the set of Mesh Portal Accounts for the user. The default value is the account name of the default account. The Name of the each key is a portal account name and the value a REG_SZ entry containing the UDF of the profile master key.

PersonalProfiles Contains the set of Mesh Profiles for the user. The default value is the UDF of the default profile master key.

The Name of each key is the UDF of the master key and the value a REG_SZ entry containing the file location of the cached copy of the personal profile.

ThisDevice Contains the set of Device profiles in the same format as the PersonalProfiles.

5.1.3. Profile data files

The profile data itself is stored in data files at the location specified in the registry. The files are standard XML files in UTF8 encoding.

5.2. OSX and Linux

All the code is regularly compiled for Linux and macOS. The unit tests are not run regularly however.

6. Implementation Status

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC6892]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.

According to [RFC6892], "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

6.1. Reference Implementation

Implementer: Phillip Hallam-Baker

Maturity: Experimental Prototype

This implementation was used to produce the reference section and all the examples in this document. Since the conversion of specification to code is automatic, there is a high degree of

assurance that the reference implementation is consistent with this document.

6.1.1. Coverage:

The draft-xx branch describes the code used to create version xx of this document.

The main current limitations are that the code only supports RSA key pairs and for ease of development the server does not persist keys across sessions. Nor does the implementation currently support the HTTP payload authentication and encryption layer or make use of TLS. These could be easily fixed.

The client and server are implemented as libraries that may be called from a multi-protocol server. A standalone server will be provided in a future release.

Only the JSON encoding is currently implemented. The JSON-B, JSON-C, ASN.1 and TLS Schema implementations are all supported by the code generation tool but not currently implemented as the build tool bindings for those encodings have not yet been finalized or documented.

The key restrictions for TLS key exchange have not yet been implemented.

The code has only been tested on Windows 10 but passed compatibility testing for both Mono and dotNetCore 2.0 run times which should in theory permit use on Linux and OSX platforms.

6.1.2. Licensing

The code is released under an MIT License

Source code is available from GitHub at https://github.com/hallambaker/Mathematical-Mesh

6.1.3. Implementation Experience

The implementation and specification documentation were developed in Visual Studio using the PHB Build Tools suite.

6.1.4. Contact Info

Contact Phillip Hallam-Baker phill@hallambaker.com

7. Security Considerations

Security Considerations are addressed in the companion document [draft-hallambaker-mesh-architecture]

8. IANA Considerations

This document specifies no actions for IANA

9. Acknowledgements

Comodo Group: Egemen Tas, Melhi Abdulhayo?lu, Rob Stradling, Robin Alden.

10. Informative References

[draft-hallambaker-mesh-architecture]
Hallam-Baker, P., "Mathematical Mesh 3.0 Part I: Architecture Guide", Work in Progress, Internet-Draft, draft-hallambaker-mesh-architecture-21, 23 October 2022, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-architecture-21>.

[draft-hallambaker-mesh-developer]
Hallam-Baker, P., "Mathematical Mesh: Reference Implementation", Work in Progress, Internet-Draft, draft-hallambaker-mesh-developer-10, 27 July 2020, <https://datatracker.ietf.org/doc/html/draft-hallambaker-mesh-developer-10>.

[PHB2017] "[Reference Not Found!]".

[RFC6892] Wilde, E., "The 'describes' Link Relation Type", RFC 6892, DOI 10.17487/RFC6892, March 2013, <https://www.rfc-editor.org/rfc/rfc6892>.

[VS2017] "[Reference Not Found!]".