Workgroup:	Internet Engineering Task Force
Internet-Draft:	draft-royer-phoenix-00
Published:	9 January 2025
Intended Status:	Informational
Expires:	13 July 2025
Author:	DM. Royer, Ed.
	RiverExplorer Games LLC

Phoenix: Lemonaid Risen Again

Abstract

Email and MIME messages account for one the largest volumes of data on the internet. The transfer of these MIME message has not had a major updated in decades. Part of the reason is that it is very important data and altering it takes a great deal of care and planning.

Another major concern is security and authentication. This proposal allows for existing autnentication to continue to work.

This is a MIME message transport that can facilitate the transfer of any kind of MIME message. Including email, calendaring, and text, image, or multimedia MIME messages. It can transfer multipart and simple MIME messages.

The POP and IMAP protocols are overly chatty and now that the Internet can handle 8-bit transfers, there is no need for the overly complex text handling of messages.

This proposal includes a sample implementation. Which also includes a gateway from this proposal to existing system. Thunderbid and Outlook plugins are part of the sample implementation.

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 13 July 2025.

Copyright Notice

Table of Contents

1. Introduction 3
   1.1. Requirements Language 4
2. Terms and Definition used in this proposal 4
3. Commands Overview 5
   3.1. Administration Commands 5
      3.1.1. Administration Commands Overview 5
   3.2. Authentication Commands 6
      3.2.1. Authentication Commands Overview 6
   3.3. Protocol 6
      3.3.1. Basic File and Folder Operations 6
   3.4. Protocol Specific Commands 7
      3.4.1. Protocol Specific Commands Overview 7
4. Over the Wire Protocol Description 7
   4.1. Over the Wire Protocol 7
      4.1.1. Over the Wire Protocol Overview 7
5. IANA Considerations 7
6. Security Considerations 7
7. References 7
   7.1. Normative References 7

7.2. Informative References 8

Appendix A. Administrative Enumerated Binary Values 8

Appendix B. Authentication Enumerated Binary Values 9

Appendix C. File and Folder Enumerated Binary Values 9

Appendix D. Protocol Enumerated Binary Values 10

Acknowledgements 11

Contributors 11

Author's Address 11

1. Introduction

On the Internet, just about everything is a MIME object and there are many ways to transport MIME. This document specifies a new application level MIME transport mechanism and protocol. This document does not specify any new or changed MIME types.

Transporting MIME objects is generally done in one of two ways: (1) Broadcasting, (2) Polling. Both methods often require some form of authentication, registration, and selecting of the desired material. These selection processes are essentially a form of remote folder management. In some cases you can only select what is provided, and in others you have some or a lot of control over the remote folders.

In addition to other functions, this specification defines a remote and local folder management. This remote folder management is common with many type of very popular protocols. This design started by looking at the very popular IMAP and POP protocols.

An additional task is transporting the perhaps very large MIME objects. Some MIME objects are so large that some devices may default to looking at only at parts of the MIME object. An example is an email message with one or more very large attachments, where the device may default to not download the large attachment without a specific request from the user.

Some objects are transported as blocks of data with a known and fixed size. These are often transported with some kind of search, get, and put commands. In effect these are folder and file commands

Other MIME objects are transported in streams of data with an unspecified size, such as streaming music, audio, or video. This specification describes how to use existing protocols to facilitate the data streaming. And again, these are folder and file commands.

A MIME object can be a simple object, or it may contain many multipart sections of small to huge size. These sections can be viewed as files in the containing MIME object.

By implementing this specification application developers can use the techniques to manage local and remote files and folders. Remote email or files are the same thing in this specification. The sections of MIME object with multipart sections are viewed as files in the MIME object. You can interact with the entire folder, or just the files withing it.

MIME object have meta data, and they are called headers. Files and folders have meta data, and they are called file attributes. This specification does not mandate any meta data, it allows for a consistent transport of existing meta data.

File and folder meta data is a complex task that can involve access control lists and permissions. This specification defines a mechanism to transport this meta data, it does not define the meta data.

And this specification provides for the ability to define both protocol extensions and the creating of finer control for specific needs that may evolve.

This examples compares current folder and file manipulations to how it can be used in this protocol with email.

- You can search for file names. You can search sender, subject, and more.
- You can search for file contents. You can search for email message contents.
- You can create, delete, and modify files. You can create, delete, and modify email messages.
- You can create, delete, and modify folders. You can create, delete, and modify email folders.

What this specification defines:

- How to use existing authentication implementations or use new ones.
- This specification describes a standard way to perform file operations that are remote to the application and agnostic to purpose of data being transported.
- Specifies a way to migrate from some existing protocols to Phoenix. Provides links to sample implementations.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terms and Definition used in this proposal

The follwing is a list of terms with their definitions as used in this specification.

ADMIN_CMD
: Administrative and auditing operations. These allow for authorized users to configure, query logs, errors, and possibly user activity.

AUTH_CMD
: Authentication and authorization operations. These operations authenticate users and verity their authorization access.

CMD
: A protocol operation. They are broken down into, ADMIN_CMD, AUTH_CMD, FILE_CMD, and PROTO_CMD.

Command
: Each packet contains a command. This command is also in the reply to the command. Not all commands have a reply. These are called a CMD.

FILE_CMD
: File and folder operations. This include creating, getting, modifying, deleting, moving, and renaming files.

Media Type
: Each MIME object has a media type that identifies the content of the object. This specification does not add, remove, or alter any MIME media type;

MIME
: This protocol transports MIME objects. This specification does not remove or alter any MIME objects;

Parameter
: Most CMD have values that are associated with it. These are called parameters. For example, the create folder CMD has the name of the new folder to be created as a parameter. When the parameter itself has values, then these CMD parameters are called PROTO_CMD, because they have their own protocol.

PROTO_CMD
: Specific protocols may have commands specific to their needs. These are calls a PROTO_CMD. An example might be extracting a specific attachment type and its data out of a MIME object. This could be a PROTO_CMD parameter to the FILE_CMD.

XDR
: RFC-4506 specifies a standard and compatible way to transfer binary information. This protocol uses XDR to transmit the CMDs and replies. The MIME data is transported as XDR opaque, as in unmodified.

3. Commands Overview

3.1. Administration Commands

3.1.1. Administration Commands Overview

Administrative command can be used to configure, audit, and manage the remote endpoint. Administrative command can be used to configure, audit, and manage user acess.

3.1.1.1. Capability Name

Implementations that support any ADMIN_CMD include ADMIN_CMD in the post authenication CAPABILITY list.

Implementations MUST NOT send a ADMIN_CMD capability in the pre authorization CAPABILITY list.

When a user has been authenticated, as part of the affermative reply, the ADMIN_CAPABILITY will be included in the reply when the user has administrative permissions.

The ADMIN_CAPABILITY reply will then be followed by the list of ADMIN commands the user is allowed to perform. For example, if a user only has permission to only view user lists, then only the USER_LIST ADMIN capibility will be provided.

3.1.1.2. Administration of users.

The following operations are defined for user administration.

Command and Capability Name	Brief Description.
USER_CREATE	Create a new user.
USER_DELETE	Delete a user.
USER_RENAME	Rename a user.
USER_LIST	List users and their capabilities.
USER_PERMISSIONS	Update user permissions.

Table 1

3.2. Authentication Commands

3.2.1. Authentication Commands Overview

TODO

3.3. Protocol

3.3.1. Basic File and Folder Operations

The file operations (FileOp) have protocol names. Here are their protocol names and a breif description.

Op Name	Brief Description.
FOLDER_CREATE	Create a new folder.
FOLDER_COPY	Copy a folder.

Op Name	Brief Description.
FOLDER_DELETE	Delete a folder.
FOLDER_RENAME	Rename a folder.
FOLDER_MOVE	Move a folder.
FOLDER_SHARE	Share a folder.
FOLDER_LIST	List folders and files.
FILE_GET	Get a known existing file.
FILE_CREATE	Create a new file.
FILE_MODIFY	Modify the contents of an existing file.
FILE_SHARE	Share a file.

Table 2

3.4. Protocol Specific Commands

3.4.1. Protocol Specific Commands Overview

TODO

4. Over the Wire Protocol Description

4.1. Over the Wire Protocol

4.1.1. Over the Wire Protocol Overview

TODO

5. IANA Considerations

This memo includes no request to IANA. [CHECK]

6. Security Considerations

This document should not affect the security of the Internet. [CHECK]

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

7.2. Informative References

[exampleRefMin] Surname [REPLACE], Initials [REPLACE]., "Title [REPLACE]", 2006.

[exampleRefOrg] Organization [REPLACE], "Title [REPLACE]", 1984, <http://www.example.com/>.

Appendix A. Administrative Enumerated Binary Values

Phoenix is a binary protocol. Each value is sent as an unsigned 32-bit integer in xdr format.

The values for the commands are arbitrary and were assigned as created. There is no plan or origination to the numbers. There is no priority or superiority to any value. The table is sorted by name, not value.

The values are not unique. They are only unique within the context in which they are used.

Some of these values are reused for other commands. For example USER_CREATE is both an (a) AUTH capability reply informing the user that they have permission to create a user with the (b) USER_CREATE command.

Some values may be reused if they are parameter arguments to other commands. For example xxxxxx.

Decimal Value	Command / Capability Name	Brief Description.
x	USER_CERT	Manage a users certificate.
x	USER_CREATE	When sent in a capability reply USER_CREATE informs the user that they have permission to create users. When sent as a command the USER_CREATE instructs the other endpoint to create a named user.
x	USER_DELETE	Delete a user.
x	USER_LIST	List users and their capabilities.

Decimal Value	Command / Capability Name	Brief Description.
x	USER_PERMISSIONS	Update user permissions.
x	USER_RENAME	Rename a user.
x	USER_RESET	Used to coordinate resetting a users authentication information.
4294967296	Reserved for future expansion.	4294967296 has a hex value of: 0xffffffff

Table 3

Appendix B. Authentication Enumerated Binary Values

Phoenix is a binary protocol. Each value is sent as an unsigned 32-bit integer in xdr format.

The values for the commands are arbitrary and were assigned as created. There is no plan or origination to the numbers. There is no priority or superiority to any value. The table is sorted by name, not value.

The values are not unique. They are only unique within the context in which they are used.

Some of these values are reused for other commands. For example USER_CREATE is both an (a) AUTH capability reply informing the user that they have permission to create a user with the (b) USER_CREATE command.

Some values may be reused if they are parameter arguments to other commands. For example xxxxxx.

Decimal Value	Command / Capability Name	Brief Description.
x	AUTH_TODO	xxx.
xxx	AUTH_xxx	xxx.
4294967296	Reserved for future expansion.	4294967296 has a hex value of: 0xffffffff

Table 4

Appendix C. File and Folder Enumerated Binary Values

Phoenix is a binary protocol. Each value is sent as an unsigned 32-bit integer in xdr format.

The values for the commands are arbitrary and were assigned as created. There is no plan or origination to the numbers. There is no priority or superiority to any value. The table is sorted by name, not value.

The values are not unique. They are only unique within the context in which they are used.

Some of these values are reused for other commands. For example USER_CREATE is both an (a) AUTH capability reply informing the user that they have permission to create a user with the (b) USER_CREATE command.

Some values may be reused if they are parameter arguments to other commands. For example xxxxxx.

Decimal Value	Command / Capability Name	Brief Description.
x	FILE_TODO	xxx.
xxx	FILE_xxx	xxx.
4294967296	Reserved for future expansion.	4294967296 has a hex value of: 0xffffffff

Table 5

Appendix D. Protocol Enumerated Binary Values

Phoenix is a binary protocol. Each value is sent as an unsigned 32-bit integer in xdr format.

The values for the commands are arbitrary and were assigned as created. There is no plan or origination to the numbers. There is no priority or superiority to any value. The table is sorted by name, not value.

The values are not unique. They are only unique within the context in which they are used.

Some of these values are reused for other commands. For example USER_CREATE is both an (a) AUTH capability reply informing the user that they have permission to create a user with the (b) USER_CREATE command.

Some values may be reused if they are parameter arguments to other commands. For example xxxxxx.

Decimal Value	Command / Capability Name	Brief Description.
x	PROTO_TODO	xxx.
xxx	PROTO_xxx	xxx.
4294967296	Reserved for future expansion.	4294967296 has a hex value of: 0xffffffff

Table 6

Acknowledgements

This template uses extracts from templates written by Pekka Savola, Elwyn Davies and Henrik Levkowetz. [REPLACE]

Contributors

Thanks to all of the contributors. [REPLACE]

Author's Address

Doug Royer (EDITOR)
RiverExplorer Games LLC
848 N. Rainbow Blvd #1120
Las Vegas, Nevada 89107
United States of America
Phone: 1+714-989-6135
Email: DouglasRoyer@gmail.com
URI: https://RiverExplorer.games