Workgroup:	Internet Engineering Task Force
Design	Phoenix-Design-00
Specification:	5 March 2025
Published:	Informational
Intended Status:	6 September 2025
Expires:	DM. Royer
Author:	*RiverExplorer LLC*

Phoenix: Lemonade Recipe

Abstract

NOTE: This is just getting started, not ready for submission yet.

This is an implementation guide for Phoenix: Lemonaide Risen Again [Phoenix].

This implementation guide, or recipe, is an example of the steps and flow of data for an implementation of a Phoenix client and server. This guide is based on a reference implementation at: https://github.com/RiverExplorer/Phoenix.

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 6 September 2025.

Copyright Notice

Table of Contents

1. Introduction

2. Definitions

Client
: The application that initiates the connection to a remote computer is called the client. The term client in this implementation guide, unless otherwise described, is referring to the client side of the Phoenix protocol, and not to any specific or general user interface.

FIFO
: A First In First Out Queue. Data is pushed into the queue at the end of the line (Queue). And the oldest data is pulled from the front of the line (Queue).

Queue
: A container for data that is like an ordered line that holds data until needed. Often referred to as a FIFO or FIFO queue.

RPC
: A Remote Procedure Call (RPC) is any method that allows a local computer to send data to a remote computer that results in the remote compute performing procedures on the behalf of the local computer. When combined with XDR it is sometimes called RPC/XDR.

  This term can be confused with a vendors specific implementation of RPC/XDR. This implementation guide is not referring to any specific implementation of RPC/XDR, it does use XDR [RFC4506] to convert data into a cross computer compatible format. And the Phoenix protocol is a form of RPC.

Server
: The application that is connected to by the client application is called the server. This guide is describing a generic Phoenix protocol server implementation.

Thread
: A Thread is somewhat like a program that run independently from the main program. Except they are within a single program. They allow the program to proceed in one area while another thread is perhaps blocked waiting for input.

  This is not to be confused with hardware threading. Some specific computer compiler may use hardware threading to accomplish this goal. Within this implementation guide, the term threading is referring to process threads and not hardware threads.

XDR
: The eXternal Data Representation or XDR, is a method of making sure that the data from one computer can be read and processed by what can be an otherwise incompatible device. See XDR [RFC4506]

[POSIX] [RFC0822] [RFC2119] [RFC4506] [RFC5234] [RFC8174] [RFC8446] [RFC9051] [rpcgendocs] [rpcgenopensource]

3. IO Model

The client and the server send data to the network, and get data from the network. This input / output model (IO Model) works with both the Phoenix client and Phoenix server.

This model support both threaded and non threaded implementations. For non-threaded systems the data is processed serially or however the implementation handles both incoming and outgoing data. The data flow is the same for both threaded and non threaded implementations.

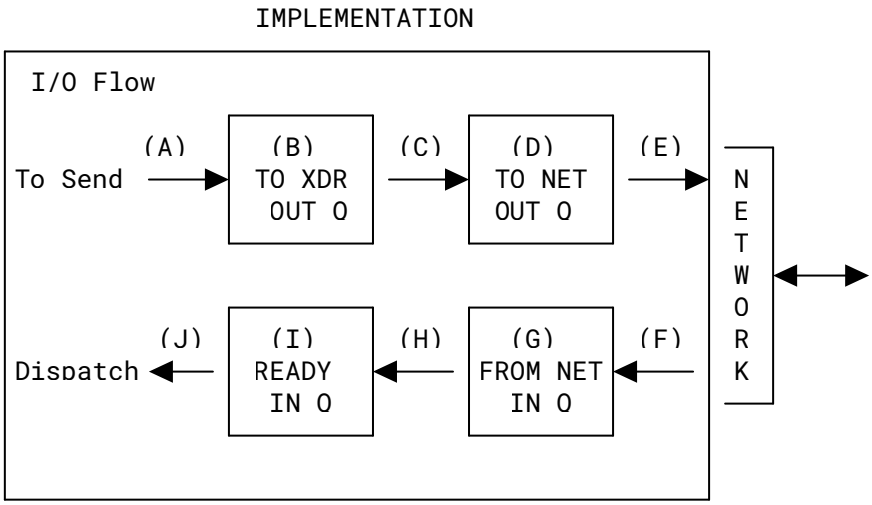

Figure 1: IO Data Flow Model - Client and Server

Where:

(A) Main Thread
: The main thread processes data and formats it into data commands and their payload to be sent to the remote endpoint. Once ready, each command is placed into the *"To XDR Out Queue"* (B).

(B) To XDR Out Queue
: This is a first in, first out queue (FIFO). Its purpose is to store data until the the *"To XDR Thread"* (C) can process the data.

(C) To XDR Thread
: This thread, takes one packet at at time out of the the *"To XDR Out Queue"* (B). Converts it from the native computer format to XDR format so it can be process by any computer architecture and stores the result into the *"To Network OUT Queue"* (D).

(D) To Network Out Queue
: This FIFO queue stores XDR encoded data until thread (E) can send it out to the network.

(E) To Network Thread
: This is the network outbound thread. It checks to make sure the network can take data. When it can, it takes one or more commands from the *"To Network Out Queue"* (D) and puts them in a PacketBody. A PacketBody may contain 1 or more commands. Then the Pack body and its one or more commands is TLS encrypted and sent out to the network.

(F) From Network Thread
This is the inbound network thread. It waits for incoming data, TLS decrypts the data.

The data arrives in PacketBody objects. A PacketBody may contain one or more commands. This thread then separates them into separate commands and stores these separate commands into the *"From Network In Queue"* (G).

(G) From Network In Queue
This is a FIFO queue. It stores incoming data packets until the XDR decode thread (H) can take them.

(H) XDR Decode Thread
This is the incoming XDR decode thread. It takes one packet at a time out of the *"From Network In Queue"* (G), converts the network binary data back into native computer format. Once the data is decoded, it paces the decoded data into the *"Ready In Queue"* (I).

(I) Ready In Queue
This FIFO queue stores inbound and decoded commands until the Dispatch Thread (J) can process the data.

(J) Dispatch Thread
This is the dispatch thread. It takes one command at a time out of the *"Ready In Queue"* (I) and dispatches it to the code that can handle the data.

4. Main Thread

The main thread in this guide, is referring to the thread assigned by the program to be the control thread. It is the essence of the application. All other threads will be referred to as worker threads.

The application that initiates the connection is called the client. The application that is connected to by the initiating application is called the server.

The functions of a client and server are not isolated. It is possible for a server to also be a client for another service. And any client could be a server to other clients. A client in this guide is referring to the client side of the protocol, and not to some specific user interface.

4.1. Client Side Main thread

A client goes through these generalized steps a startup:

1. The client main thread starts up and reads any implementation specific configuration information. The configuration information will include zero or more sets of server data needed to contact Phoenix servers.

   In the case when zero servers are configured, the program waits for server configuration information, perhaps from a user.

The client may at any time acquire additional server connection information and add them to the list of valid servers.

2. The client starts all worker threads and waits for them to signal they are ready.
3. When the worker threads are ready the client uses the configuration data and initiates a connection to each configured server according to the Phoenix protocol specification. See Initial Connection to Server (Section 10).
4. As each server connection is established, the client starts its specific tasks. Any client may support multiple tasks. These could include email, calendaring, network news, RSS feeds, remote administrative control, or any other tasks designed into the client and server implementations.

   In the case of an email client, it would start sending folder open packets to the server to gather emails.

   A calendaring client would look for new calendar entries or processing requests. It would send any pending calendaring request to the server. And synchronize differences according to calendaring specifications.

   Some servers could be bulletin boards or news feeds with read only access. Other server might be submission or data storage servers.

   As each task is determined, the data is gathered into A Phoenix command with its payload and stored into the To XDR Queue (B) as shown in the IOModel (Figure 1)

4.2. Server Side Main thread

5. To XDR Thread

6. To Network Thread

7. From Network Thread

8. XDR Decode Thread

9. Dispatch Thread

10. Initial Connection to Server

11. IANA Considerations

TODO

12. Security Considerations

13. References

13.1. Normative References

[Phoenix] Royer, DM., "Phoenix: Lemonade Risen Again", February 2025, <https://htmlpreview.github.io/?https://github.com/RiverExplorer/Phoenix/blob/main/Documentation/draft-royer-phoenix.html>.

[POSIX] IEEE, "1003.1-2024 - IEEE/Open Group Standard for Information Technology--Portable Operating System Interface (POSIX™) Base Specifications", June 2024, <https://ieeexplore.ieee.org/servlet/opac?punumber=10555527>.

[RFC0822] Crocker, D., "STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES", STD 11, RFC 822, DOI 10.17487/RFC0822, August 1982, <https://www.rfc-editor.org/info/rfc822>.

[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>.

[RFC4506] Eisler, M., Ed., "XDR: External Data Representation Standard", STD 67, RFC 4506, DOI 10.17487/RFC4506, May 2006, <https://www.rfc-editor.org/info/rfc4506>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[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>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC9051] Melnikov, A., Ed. and B. Leiba, Ed., "Internet Message Access Protocol (IMAP) - Version 4rev2", RFC 9051, DOI 10.17487/RFC9051, August 2021, <https://www.rfc-editor.org/info/rfc9051>.

13.2. Informative References

[PhoenixImplementation] Royer, D., "Phoenix Sample Implementation", 2025, <https://github.com/RiverExplorer/Phoenix>.

[rpcgendocs] Unknown Author, "rpcgen Protocol Compiler", January 2025, <https://docs.oracle.com/cd/E37838_01/html/E61058/rpcgenpguide-12915.html>.

[rpcgenopensource] Unknown Author, "rpcgen++ Open Source Tool", January 1983, <https://github.com/RiverExplorer/Phoenix/tree/main/rpcgen%2B%2B-src>.

Acknowledgments

Contributors

Thanks to all of the contributors. [REPLACE]

Author's Address

Doug Royer
RiverExplorer LLC
848 N. Rainbow Blvd #1120
Las Vegas, Nevada 89107
United States of America
Phone: 1+208-806-1358
Email: DouglasRoyer@gmail.com
URI: https://DougRoyer.US