

Project Goals for Ocamlnet

Patrick Doane

March 14, 2001

The Ocamlnet project will provide a collection of modules for the Objective Caml language which focus on application-level Internet protocols and conventions. In addition to this software deliverable, there are several motivational questions we want to explore.

1. How can volunteers coordinate in a structured fashion to create quality software?
2. What practices enable a part-time development staff to follow a software process?
3. What effect do the Internet and open source development have on achieving these goals?

Since many of the components of this project are independent, we will establish an initial software process that will be refined throughout the lifetime of the project.

Our API will match existing specifications and proposals as closely as possible. This should allow the implementation to serve as a reference for other developers and as an environment to explore extensions and modifications. However, common operations will also be provided to make the library easy and approachable for beginners. For protocols that are particularly complex, we may want to explore the design of a higher-level interface. To validate these goals, the distribution will include several example applications. These should vary in complexity and scope, to illustrate various uses of the modules.

The target platforms for use (and development) of Ocamlnet will be Unix systems. The code should work properly on any system which fully implements the Unix module provided by the Ocaml distribution. This may extend to include Windows and OS X but this is not a requirement.

The first section of this document enumerates various Internet protocols we hope to support and is followed by a section describing the manipulation of data associated with these protocols. The last section contains information about related work.

1 Protocols

This section describes the application-level protocols that will be implemented for Ocamlnet. Most of these protocols are specified in IETF published RFCs and standard documents. The API will focus initially on the client-side of the protocols and may delay the implementation of servers until a second phase of development. It may, however, be useful to implement simple servers for testing the client API.

ACAP

The Application Configuration Access Protocol (ACAP) is designed to store user data, such as preferences, at a remote location. The protocol makes it easy to add new information and standardize on data formats. It also supports access control lists to restrict information .

The implementation will adhere to RFC 2244 [1].

CGI

The Common Gateway Interface (CGI) is a simple interface for running external programs on an HTTP server in a platform-independent manner.

The implementation will adhere to the IETF draft 3 [2].

FTP

The File Transfer Protocol (FTP) provides a reliable and efficient means for transmission of files involving a remote computer. Programs using FTP do not have to worry about the file system details among hosts as the protocol works independently of variations in file systems.

The implementation will adhere to RFC 0959 [3] and the following extensions.

- FTP Security Extensions (RFC 2228 [4])
- Internationalization of the File Transfer Protocol (RFC 2640 [5])

HTTP/1.1

The Hypertext Transfer Protocol (HTTP) provides a mechanism for the distribution of hypermedia information systems. It is commonly used for the distribution of HTML documents, however systems can use HTTP for a variety of data formats.

The implementation will adhere to RFC 2616 [6] and the following extensions.

- HTTP Authentication RFC 2617 [7].
- HTTP State Management Mechanism RFC 2965 [8].

IMAP4rev1

The Internet Message Access Protocol, Version 4rev1 (IMAP4rev1) provides a powerful mechanism for clients to manipulate an electronic mail store on a server. It supports the notion of hierarchical message folders and can be used in a mixed e-mail/newsgroup environment. Novel features include the ability for an off-line client to synchronize with a server allowing disconnected use.

The implementation will adhere to RFC 2060 [9] and will consider many of the extensions proposed for IMAP. It is expected that most of the extensions provided by the Cyrus mail server will be implemented.

ICQ

ICQ is a protocol for online chat similar to Instant Messages on America Online.

This implementation will follow the unofficial specification by Henrik Isaksson [10].

IRC

The IRC (Internet Relay Chat) protocol enables text-based discussion for users grouped in forums, called channels. These channels are similar to chat rooms on America Online.

This implementation will adhere to the following RFC documents.

- Server Protocol (RFC 2813 [11])
- Client Protocol (RFC 2812 [12])
- Channel Management (RFC 2811 [13])
- Architecture (RFC 2810 [14])

NNTP

Network News Transfer Protocol (NNTP) specifies a protocol for the distribution, inquiry, retrieval, and posting of news articles.

The implementation will adhere to RFC 0977 [15] and the Common NNTP Extensions (RFC 2980 [16]).

POP3

The Post Office Protocol - Version 3 (POP3) is a common protocol for a workstation to retrieve mail from a remote server. POP3 does not provide extensive manipulation of mail on the server; normally, mail is downloaded and then deleted. Users wanting more power and flexibility should consider using IMAP.

The implementation will adhere to RFC 1939 [17] and the following extensions.

- POP3 AUTHentication command (RFC 1734 [18])
- POP3 Extension Mechanism (RFC 2449 [19])

SMTP

The Simple Mail Transfer Protocol (SMTP) provides a means for reliable and efficient transfer of mail messages.

The implementation will adhere to RFC 0821 [20] from the perspective of a Message User Agent (see RFC 2476 [21]) and will consider the following extensions.

- SMTP Service Extensions (RFC 1869 [22]).
- 8-bit MIME Transport (RFC 1652 [23])
- Binary Chunking (RFC 3030 [24])
- Message Size Declaration (RFC 1870 [25])
- Delivery Status Notification (RFC 1891 [26])
- Enhanced Mail System Status Codes (RFC 1893, RFC 2034 [27, 28])
- Command Pipelining (RFC 2197 [29])

TELNET

The telephone networking (TELNET) protocol provides a flexible means to run remote terminal-based applications. The protocol makes extensive use of negotiations, allowing both client and server to communicate their capabilities and agree on how a session will operate. The protocol also defines some simple control sequences, including querying a server's "alive" status ("AYT"), and interrupting a remote process ("IP").

The implementation will adhere to RFC 0854 [30].

TELNET can be extended with little effort, and there are some typical extensions that clients do, including:

- go-ahead suppression
- “line mode”

The Ocamlnet project may implement some of these extensions; the reason to implement an extension is typically application-specific (so you don’t implement it unless you really need it for something).

2 Data Manipulation

There are several data formats and string manipulations that are commonly used across different protocols on the Internet, including:

- URL parsing
- Base64 encoding
- Quoted printable encoding
- Mail messages / MIME
- BinHex
- Uuencoding

The scope and detail of functionality in this section will evolve throughout the implementation of Ocamlnet. Also, it has not been decided if parsers should be provided for common file formats such as mailcap or message boxes.

3 Related Work

There are a few libraries which have already been released for Objective Caml that meet some of the project goals.

- The `netstring` library by Gerd Stolpmann contains a number of string processing functions that supports many of the features in section 2. Stolpmann has also released the `netclient` library as an HTTP 1.1 client. It will be worthwhile to see how these efforts can collaborate.
- The `cgi` library by Jean-Christophe Filliatre is a simple API that has been enhanced as part of the `net-string` library.
- The `wserver` library by Daniel de Rauglaudre can be used for writing simple web servers.

Since there are many libraries written in C that satisfy some of the project goals, one might argue for writing Ocaml wrappers to these existing libraries. If our only goal was to provide a usable library, then this would be a sensible approach. However, this would limit our ability to investigate the issues that will come about through the design and implementation of this software.

4 Acknowledgements

Thanks to Kevin Grant, Lyren Brown, and Ruth Doane for contributing, reviewing, and commenting on this document.

License

This documentation is provided by the Ocamlnet project “as is” and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Ocamlnet project be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this documentation, even if advised of the possibility of such damage.

Revision History

Date	Version	Description	Responsible Person
February 13, 2001	draft 1	First draft for comments and review	Patrick Doane
March 14, 2001	1.0	Updated based on review comments	Patrick Doane

References

- [1] C. Newman and J. G. Myers. RFC 2244: ACAP — Application Configuration Access Protocol, November 1997.
- [2] Ken A L Coar. The www common gateway interface version 1.1 (draft 3), June 1999.
- [3] J. Postel and J. K. Reynolds. RFC 959: File transfer protocol, October 1985.
- [4] M. Horowitz and S. Lunt. RFC 2228: FTP security extensions, October 1997.
- [5] B. Curtin. RFC 2640: Internationalization of the file transfer protocol, July 1999.
- [6] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protol – http/1.1, June 1999.
- [7] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach, A. Luotonen, and L. Stewart. Http authentication: Basic and digest access authentication, June 1999.
- [8] D. Kristol and L. Montulli. RFC 2965: Http state management mechanism, October 2000.
- [9] M. Crispin. RFC 2060: INTERNET MESSAGE ACCESS PROTOCOL — VERSION 4rev1, December 1996.
- [10] Henrik Isaksson. Version 5 of the icq protocol, December 2000.

- [11] C. Kalt. RFC 2813: Internet relay chat: Server protocol, April 2000.
- [12] C. Kalt. RFC 2812: Internet relay chat: Client protocol, April 2000.
- [13] C. Kalt. RFC 2811: Internet relay chat: Channel management, April 2000.
- [14] C. Kalt. RFC 2810: Internet relay chat: Architecture, April 2000.
- [15] B. Kantor and P. Lapsley. RFC 977: Network news transfer protocol: A proposed standard for the stream-based transmission of news, February 1986.
- [16] S. Barber. RFC 2980: Common nntp extensions, October 2000.
- [17] J. Myers and M. Rose. RFC 1939: Post Office Protocol — version 3, May 1996.
- [18] J. Myers. RFC 1734: POP3 AUTHentication command, December 1994.
- [19] R. Gellens, C. Newman, and L. Lundblade. RFC 2449: POP3 extension mechanism, November 1998.
- [20] J. Postel. RFC 821: Simple mail transfer protocol, August 1982.
- [21] R. Gellens and J. Klensin. RFC 2476: Message submission, December 1998.
- [22] J. Klensin, N. Freed, M. Rose, E. Stefferud, and D. Crocker. RFC 1869: SMTP service extensions, November 1995.
- [23] J. Klensin, N. Freed, M. Rose, E. Stefferud, and D. Crocker. RFC 1652: SMTP service extension for 8bit-MIMEtransport, July 1994.
- [24] G. Vaudreuil. RFC 3030: Smtp service extensions for transmission of large and binary mime messages, December 2000.
- [25] J. Klensin, N. Freed, and K. Moore. RFC 1870: SMTP service extension for message size declaration, November 1995.
- [26] K. Moore. RFC 1891: SMTP service extension for delivery status notifications, January 1996.
- [27] G. Vaudreuil. RFC 1893: Enhanced mail system status codes, January 1996.
- [28] N. Freed. RFC 2034: SMTP service extension for returning enhanced error codes, October 1996.
- [29] N. Freed. RFC 2197: SMTP service extension for command pipelining, September 1997.
- [30] J. Postel and J. K. Reynolds. RFC 854: Telnet Protocol specification, May 1983.