H. Sugano
S. Fujimoto
Fujitsu

F. Mazzoldi
A. Diacakis
Personity, Inc.

G. Hudson
MIT

J. D. Ramsdell
The MITRE Corporation

Expires: April 2002 October 2001

Presence and Instant Messaging Protocol (PRIM) Server-Server Protocol Specification <draft-ietf-prim-server-00.txt>

Status of this Memo

Copyright Notice

Abstract

The architecture and specifications of the Presence and Instant Messaging protocols (PRIM) are described. PRIM defines a set of protocols for the Presence and Instant Messaging services which satisfy the IMPP requirements [RFC2779]. PRIM is also designed so as to conform with the Common Profile for Instant Messaging (CPIM) specification being developed in the IMPP WG. This memo describes the PRIM Server-Server protocol specification.

Table of Contents

1. Introduction

Instant Messaging and Presence (IM/P) services provide users a way to know others are available to communicate with them primarily by exchanging short text messages and possibly by other communications media such as voice and/or video. The PRIM, PResence and Instant Messaging, protocols are designed for such services so that these services can be provided by a set of servers distributed across a large number of administrative domains.

PRIM specifications are classified into two parts; a client-server protocol specification and a server-server protocol specification. The former is the protocol for clients of the PRIM IM/P services to communicate with the IM/P servers exchanging PRESENCE INFORMATION and INSTANT MESSAGES, and it is mainly used within a single administrative domain. On the other hand, the latter is the protocol for the IM/P servers to communicate with other servers possibly in the different domains. This separation is meaningful because of the simplified architecture of PRIM described below.

This memo gives the PRIM server-server protocol specification. This serves as a protocol not only for communications between the PRIM servers, but also for communications between different domains that may internally use other protocols than PRIM. This is accomplished by gatewaying the internal protocol to the PRIM protocol. The specification of the PRIM client-server protocol is presented in a separate document.

The PRIM specifications are developed on the basis of the IMPP activities such as the Model and Requirements documents for the IM/P services [RFC2778,RFC2779]. PRIM is also designed to conform to the Common Profile for Instant Messaging (CPIM) specifications being developed by the IMPP WG. This enables that users of PRIM services exchange PRESENCE INFORMATION and INSTANT MESSAGES with the users of the services which use other CPIM compatible protocols.

1.1. Design Principles

Some of the design principles on which the PRIM specifications are based are as follows. Note that the latter two are only relevant to the PRIM client-server protocol.

o Transfer protocol directly atop of TCP

PRIM assumes TCP as the basic transport mechanism for INSTANT MESSAGES and PRESENCE INFORMATION. TCP provides a sufficiently reliable transport infrastructure which is required by both INSTANT

MESSAGING and PRESENCE SERVICES.

o Long-lived Client/Server connections

PRIM uses long-lived client/server TCP connections in order to receive INSTANT MESSAGES and PRESENCE INFORMATION NOTIFICATIONS. Note that this is the prevailing model used by most Presence and IM systems today. It brings the following advantages:

- Overhead is reduced, because authentication is performed once, at the beginning of the connection. This is important, for example, when PRESENCE INFORMATION NOTIFICATIONS occur frequently.

- Connections are firewall friendly, because USER AGENTS initiate connections from inside a firewall that can carry NOTIFICATIONS or messages initiated from the outside.

o Selective Presence Publication

[RFC2779] stipulates various requirements for access control; 2.3.x and several in section 5. Among others, we consider the feature of "Polite Blocking" (5.1.15, 5.2.3) to be very important for PRESENCE SERVICES. This protocol contains a mechanism for such selective PRESENCE INFORMATION publication as well as in-band access control.

1.2. Terminology

[RFC2778] and [RFC2779] define the terminology for the PRESENCE and INSTANT MESSAGING fields. Please refer to those documents for a complete glossary of the UPPER CASED terms.

The key words "MUST", "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [34].

2. Architecture

This section describes an overall PRIM architecture for both client-server and server-server protocols.

2.1. Overall Architecture

The PRIM architecture involves two components: Service Domains and USER AGENTS. A Service Domain in the context of PRIM is an administrative entity where a PRINCIPAL has its identifier as an entity such as PRESENTITY/WATCHER and SENDER/INBOX to enjoy the PRESENCE and INSTANT MESSAGING SERVICES. A PRINCIPAL's Service

Domain is called its Home Domain, and the PRINCIPAL connects to its Home Domain via a USER AGENT to access PRESENCE and INSTANT MESSAGING SERVICES.

A Service Domain consists of PRESENCE and/or INSTANT MESSAGING SERVERS together with USER AGENTS. A USER AGENT only communicates with the SERVERS in its HOME DOMAIN, and the PRIM client-server protocol is used by the USER AGENT and SERVERS. A SERVER can communicate with other SERVERS using the PRIM server-server protocol specified in this memo. The protocol commands transferred by this protocol are those either initiated by the SERVER itself or relayed on behalf of USER AGENTS. These SERVERS may be located in different Service Domains.

The PRIM protocols are both connection-based, i.e. every protocol commands are transferred through a TCP connection. A USER AGENT communicating with a SERVER exchanges the protocol commands through a client-server connection established between the SERVER and the USER AGENT. Similarly, a SERVER communicating with another SERVER exchanges the protocol commands using a server-server connection between the two SERVERS.

2.2. Presence Model

2.2.1. Presence Servers

Presence servers are primary components of PRESENCE SERVICE. A presence server in a Service Domain stores and manages PRESENCE INFORMATION published by PRESENTITIES in that domain and SUBSCRIPTIONS from SUBSCRIBERS to the PRESENTITIES. The SUBSCRIBERS may be located in the same domain and may subscribe from different domains. If a presence server receives a request for a PRESENTITY in a different domain, it forwards the request to the target domain using an inter-domain server-server connection.

When a part of PRESENCE INFORMATION of a PRESENTITY is changed, NOTIFICATION messages for relevant SUBSCRIBERS to that particular PRESENTITY will be issued by the presence server.

2.2.2. Presence Subscriptions

WATCHERS can subscribe to a PRESENTITY in order to receive NOTIFICATIONS when the PRESENCE INFORMATION of that PRESENTITY changes.

SUBSCRIPTIONS have a duration under which they are in effect. This duration is specified at the time that the subscription is placed or

renewed. Once that period elapses, the SUBSCRIPTION has to be either renewed by the SUBSCRIBER, or else it MUST be removed by the PRESENTITY's Presence Server.

This renewal may be either issued by the USER AGENT, or by the SUBSCRIBER's Presence Server on behalf of the SUBSCRIBER.

2.2.3. PRESENCE INFORMATION

PRESENCE INFORMATION transported by the PRIM protocol consists of one or more PRESENCE TUPLEs, as defined by the IMPP model document [RFC2778]. PRIM adopts the CPIM Presence Information Data Format [CPIM-PIDF] as its presence data format.

2.3. Instant Messaging Model

INSTANT MESSAGING SERVICE provides a functionality of sending and receiving INSTANT MESSAGES for PRINCIPALS. USER AGENTS are able to exchange INSTANT MESSAGES with a client-server connection to the INSTANT MESSAGING SERVERS. An INSTANT MESSAGING SERVER provides an INSTANT INBOX for receiving Instant Messages.

When a USER AGENT wishes to start receiving INSTANT MESSAGES, it starts listening to that INSTANT INBOX. Conversely, when it no longer wishes to receive INSTANT MESSAGES from that INSTANT INBOX, it stops listening to the INBOX.

INSTANT INBOXes have two states, as described in RFC 2779: OPEN and CLOSED. An INBOX is OPEN when at least one PRINCIPAL is listening to that INBOX. It is CLOSED when there are no PRINCIPALS listening to the INBOX.

If an INSTANT MESSAGE is sent to an INBOX that has multiple PRINCIPALS listening, the message is considered to be delivered successfully if at least one PRINCIPAL receives it.

3. Identifier Namespace

This section defines the syntax of identifiers which appear as the protocol elements. The ABNF [RFC 2234] is used for the syntax definitions.

The next ABNF defines a Presence or IM identifiers, which are used to identify PRESENTITIES and INSTANT INBOXes respectively. It also defines IP address formats to be referred in some header definitions.

```
presence-id     = word-pres ":" local-part "@" domain
im-id           = word-im ":" local-part "@" domain
local-part      = 1*( unreserved / escaped )
unreserved      = ALPHA / DIGIT / "!" / "$" / "&" / "'" / "*"
                / "." / "+" / "-" / "/" / "=" / "?" / "_" / "~"
escaped         = "%" hex-char hex-char
hex-char        = DIGIT / "A" / "B" / "C" / "D" / "E" / "F"
                / "a" / "b" / "c" / "d" / "e" / "f"
domain          = 1*domain-label *("." 1*domain-label)
domain-label    = 1*( unreserved / escaped )
word-pres       = %x70.72.65.73     ; "pres"
word-im         = %x69.6D           ; "im"
decimal-byte    = 1*3DIGIT
ALPHA           = <defined by RFC 2234 -- 'A'-'Z' / 'a'-'z'>
DIGIT           = <defined by RFC 2234 -- '0'-'9'>

hex4            = 1*4hex-char
hexseq          = hex4 *(":" hex4)
ip6-address     = hexseq / hexseq "::" [ hexseq ] / "::" [ hexseq ]
ip4-address     = "::" 1*1decimal-byte 3*3("." 1*1decimal-byte)
```

The PRIM Presence and IM identifiers are defined so as to align with CPIM [CPIM]. They have the form of URI [RFC2396] and the same URI schemes are selected for Presence identifiers ("pres:") and IM identifiers ("im:").

The syntax for the "local-part" and "domain" of those identifiers are similar to that for email addresses, specified as addr-spec in [RFC822]. But, the characters defined in this specification is restricted so as to conform to the URI syntax [RFC2396]. The characters which are not allowed in this definition MUST be escaped. Also note that, unlike a mailto: URL [RFC 2368], a pres: or im: URL cannot contain multiple addresses.

Moreover, The syntax for "domain-label" here is so defined that it will be conformant to the prospective specification of the Internationalized Domain Name [IDN]. A string for "domain" MUST be a valid domain name according to the rules currently in existence.

Followings are some examples of valid Presence and IM identifiers:

```
pres:joe@example.net
im:%22Jane%20Smith%22@domain.com
```

A PRIM USER AGENT SHOULD recognize a PRESENTITY or INSTANT INBOX identifier without the scheme if it is entered in a PRESENCE or INSTANT MESSAGING context. Similarly, a USER AGENT SHOULD display a PRESENCE or INSTANT MESSAGING identifier without the scheme if it is

displayed in a PRESENCE or INSTANT MESSAGING context.

A PRINCIPAL may or may not have the same IDENTIFIER for its PRESENTITY and its IM INBOX. However, for an integrated Presence and IM service, the service SHOULD NOT assign the IDENTIFIERS which are different only in the scheme part to different PRINCIPALS.

4. Establishing Connections

4.1. Server-server Connections

A Presence or Instant Messaging Server send a command to another server through a server-server connection. The command may be the one issued by the server itself or the one issued originally by a USER AGENT and forwarded. It can reuse an existing connection to the destination server if already exists. If there is no connection to the destination, the originating server tries to establish a new connection. To do that, it will resolve the name of the recipient of the command to locate the destination server.

When a server establishes a connection to another server, that connection end-point can be authorized to communicate on behalf of multiple PRESENTITIES or INBOXES. For example, if server A receives a subscription request from server B, on behalf of user thanos@personity.com, server A MUST verify that server B is one of the servers of the personity.com domain. If so, it will then accept other requests from server B that pertain to users of the personity.com domain.

PRIM provides several methods to authenticate and authorize servers, which are described in section 1x.x.

The connection may be closed by either side at any time when there are no outstanding commands on the connection from that server's point of view. A server which has received a command may close the connection if it encounters a serious error during the processing of the command. In this case, the server SHOULD respond with "400 Bad Request" error if possible before closing the connection. Any commands sent to a server which closed the connection before sending a reply can safely be assumed to have gone unprocessed.

4.2. Connections and Services

PRIM specifications allow the separation of PRESENCE SERVICE and INSTANT MESSAGING SERVICE, i.e. the specifications allow a server only providing one of those services. A server (and a USER AGENT as

well) may establish two distinct connections for the two services even though they are provided by the same domain. Of course, a server which serves only one of these services may establish only one connection for that service to a single domain.

4.3. Name Resolution

For the server location, PRIM reuses the existing Domain Name Services to achieve this. The domain name resolution is performed with the domain name of the destination address of the PRIM command to be transported.

A server MUST discover a remote domain's server using the following algorithm: the server performs a SRV [RFC 2782] lookup for the remote domain using the protocol "tcp" and the service "prim-pr" (for PRESENCE) or "prim-im" (for INSTANT MESSAGING). If the two SRV lookups for the "prim-pr" and "prim-im" services in a domain return the same host and port number, the server MAY establish a single connection to that host/port to enjoy the two services.

If no SRV record is present, the server performs an A lookup on the remote domain and uses the resulting IP addresses with the allocated port [xxx] for PRESENCE or [xxx] for INSTANT MESSAGING.

Note: The protocol is capable of using two different TCP ports: one for the PRESENCE SERVICE and one for the INSTANT MESSAGING SERVICE. However, the usage of one or two ports will be possible for different needs. The protocol ensures there is no ambiguity between commands received from different services.

4.4. Shared Connections

When a domain provides both the PRESENCE and INSTANT MESSAGING SERVICES in a single host and port, it has been declared using the DNS SRV RR as stated in the previous section. In that case, the initiating server MAY open a single connection and authenticate itself once on that connection using one of available authentication methods.

The server can differentiate between the presence and instant messaging service commands by the command name itself. The "general" commands such as LOGIN, STARTTLS, or PING do not care which services they are used for in a shared connection. If the STARTTLS command is needed for the connection, one STARTTLS command is sufficient.

If a server received a command for the service it does not provide for some unexpected reason, the server MUST respond with '501 Not Implemented' error.

5. Command Structure

This section describes the structure of generic PRIM commands and also gives a classification of PRIM requests based on the connections on which they are transported. The details of the requests specifications are described separately in the later sections.

5.1. Generic Commands

A connection transports a sequence of commands. The underlying character set for commands is Unicode, represented in UTF-8 [RFC 2279]. Command bodies are an exception; they should be treated as unprocessed octets. An implementation MUST properly handle arbitrary binary data in the body. A command is either a request or a response.

```
PRIM-command = request / response
```

Requests and responses use the generic command format of [RFC822] for transferring entities (the body of the command). Both types of command consist of a start-line, one or more command-header fields (also known as "headers"), an empty line (i.e., a line with nothing preceding the CRLF) indicating the end of the header fields, and an optional command-body.

```
generic-command = start-line
                  *command-header
                  CRLF
                  [ command-body ]

     start-line = request-line / response-line
```

Receivers of commands SHOULD ignore any empty line(s) received where a start-line is expected.

5.1.1. Command Headers

PRIM command-header fields follow the same syntactic restriction as specified by [CPIM-MSGFMT]. Thus, each header field consists of a header name followed by a colon ("%x3A"), a single whitespace ("%x20") and a field value. Header names are case-sensitive. The entire header MUST be contained in a single line.

Command header fields are categorized into four types; general-header, presence-header, im-header, and entity-header. General-header fields are applicable to both of PRIM Presence and Instant Messaging Protocols, and used for controlling the basic behavior of the PRIM applications, such as connection management and delivery of

the commands. Presence-header fields and im-header fields are included as a meta-data of the content of the commands of the Presence and Instant Messaging Protocols respectively. Entity-header fields describes the common feature of the body of the command.

```
command-header = (general-header     ;
                  / presence-header  ;
                  / im-header        ;
                  / entity-header    ;
                  )
```

5.1.2. Command Body

Some commands of the PRIM Presence and Instant Messaging Protocols can contain a command-body. The command-body is used to carry presence information, instant message, or other information.

5.2. Requests

A generic PRIM request includes the method to be applied to the resource, the protocol version, and the data needed for asynchronous requests.

```
request-line = method
               SP version
               SP request-identifier
               SP content-length
               CRLF
```

As PRIM specifies two kinds of protocols for Presence and Instant Messaging Services, a PRIM request is classified into two categories.

```
request = (presence-request / im-request)

presence-request = request-line
                   *((general-header
                    / presence-header
                    / entity-header) CRLF)
                    CRLF
                    [ command-body ]

im-request = request-line
             *((general-header
              / im-header
              / entity-header) CRLF)
```

```
                        CRLF
                        [ command-body ]
```

5.2.1. Method

The method token indicates the method to be performed on the resource. Here, methods are categorized into three groups; presence methods for presence services, IM methods for instant messaging services, and general methods for both. In section 8.4, these are further classified for the detailed specifications.

```
              method = general-method
                     / presence-method
                     / im-method

      general-method = "LOGIN"              ; Section 7.3.1
                     / "STARTTLS"           ; Section 7.3.2
                     / "LOGOUT"             ; Section 7.3.3
                     / "PING"               ; Section 7.3.4
                     / "VERIFYSERVER"       ; Section 7.3.5

     presence-method = "SUBSCRIBE"          ; Section 7.1.1
                     / "UNSUBSCRIBE"        ; Section 7.1.2
                     / "NOTIFY"             ; Section 7.1.3

          im-method = "SEND"                ; Section 7.2.1
```

5.2.2. Version

The version identifies the version of the protocol in use. It contains the name string specifying the protocol and the major and minor version numbers.

```
              version = "PRIM" "/" 1*DIGIT "." 1*DIGIT
```

PRIM adopts the similar protocol versioning policy to those described in RFC 2145 [RFC2145] and RFC 2616 [HTTP1.1]. Thus, the protocol version is intended to allow the sender to indicate the format of a command and its capability for understanding further communication. See RFC 2616 and 2145 for more detailed explanations.

A PRIM application SHOULD send a command version equal to the highest version for which it is at least conditionally compliant, and whose major version is no higher than the highest version supported by the other end, if it is known. A PRIM application MUST NOT send a version for which it is not at least conditionally compliant.

A server MAY send a 505 (Version Not Supported) response if cannot send a response using the major version used in the client's request.

5.2.3. Request Identifier

Request identifiers are used to implement asynchronous requests.

```
request-identifier = 1*[ALPHA / DIGIT] / "-"
```

An endpoint of a connection is responsible for generating request identifiers, and the request identifiers are used to match responses it receives with the requests it has sent. The other endpoint of a connection is responsible for labeling a response with the identifier it received in the request. An identifier may be reused after the endpoint receives the response to the request with the identifier.

The request identifier of a command is "-" if and only if the request expects no reply. If an endpoint receives a request with the request identifier "-", it MUST NOT send any response to the request.

5.2.4. Content Length

The content-length header contains the length of the command body in bytes.

```
content-length = 1*DIGIT
```

5.3. Responses

A response includes many of the same fields as a request with the addition of a status code and a response phrase.

```
response-line = version
                SP request-identifier
                SP content-length
                SP status-code
                SP response-phrase
                CRLF
```

The request identifier in the response MUST NOT be "-".

The status-code is a 3 digit code and the response-phrase is a short message description. The values are defined in Section 8.

Some status codes are common to all commands, whereas others are only

used by a subset of commands. Common status codes to all commands are:

```
200 OK
300 Redirect
400 Bad Request
401 Unauthorized (except for LOGIN)
402 Forbidden (except for LOGIN, STARTTLS, PING)
501 Internal Server Error
503 Version Not Supported
```

6. Command Headers

Command headers are defined as follows:

```
general-header = (from-header
                  / to-header
                  / auth-state-header
                  / SASL-mechanism-header
                  / redirect-header
                  / server-address-header
                  / astrength-header
                  / date-header
                  )

presence-header =  duration-header

im-header = (message-id-header
                  / conversation-id-header
                  / reply-to-header
                  )

entity-header =  content-type-header
```

6.1. General Headers

6.1.1. From

Identifies the PRESENTITY or INBOX that issued this command, or that it was issued on behalf of.

```
from-header = "From: " ( presence-id / im-id )
```

The value of this header is either "presence-id" or "im-id". "presence-id" MUST be used only if the underlying method is a

"presence-method" or "general-method". "im-id" MUST be used only if the underlying method is an "im-method" or "general-method".

The receiving end of a command SHOULD always check that the sender is authorized to send commands on behalf of the identifier in the from-header, as described in Sections 13.

6.1.2. To

Specifies the PRESENTITY or INBOX this command is intended to.

```
to-header = "To: " ( presence-id / im-id )
```

The value of this header is either "presence-id" or "im-id". "presence-id" MUST be used only if the underlying method is a "presence-method" or "general-method". "im-id" MUST be used only if the underlying method is an "im-method" or "general-method".

6.1.3. Domain

Identifies the domain when used with the LOGIN command in the server-server connection.

```
domain-header = "Domain: " domain
```

The value of this header MUST be a valid domain name.

6.1.4. Auth-State

Indicates the status in the authentication process in the LOGIN command.

```
auth-state-header = "Auth-State: "
                    ( "init"
                    / "continue"
                    / "abort" )
```

6.1.5. SASL-Mechanism

Specifies the SASL mechanism in the LOGIN request or the response to the LOGIN request. In the request, the SASL mechanism the USER AGENT wants to use MUST be specified. When used in the response, one or more mechanisms which the server supports MAY be specified.

```
SASL-mechanism-header = "SASL-Mech: " mechanisms
           mechanisms = mechanism [ *(SP mechanism) ]
            mechanism = 1*20(ALPHA / DIGIT / "-" / "_")
```

6.1.6. Redirect

When a server cannot handle requests from a USER AGENT or other server, it issues an error response "300 Redirect" which includes the redirect-header. This lets the caller know that its request cannot be handled at this server and an alternative server address and port are provided.

```
redirect-header = "Redirect: " address SP port
        address = domain / ip4-address / ip6-address
           port = 1*DIGIT
```

6.1.7. Server-Address

Indicates the IP address for the server that is initiating the connection. This header is used in the VERIFYSERVER method to show the address of the server that needs verification (see Sections 10.5 and 13.2).

```
server-address-header = "Server-Address: "
                        ( ip4-address / ip6-address )
```

6.1.8. AStrength

When a server acts as a relay, it MUST communicate to the next node a rough indication of the authentication strength of the previous hops using the "Astrength" header unless the servers already have knowledge about the relaying server's authentication strength through some out-of-band manner. If the servers acknowledge the authentication strength of the other end, the AStrength header MAY NOT appear in the relayed request.

The syntax for the Astrength header is:

```
astrength-header = "AStrength: " astrength
       astrength = "strong" / "medium" / "weak" / "none"
```

The meanings of the astrength values are:

strong	Command authenticity and integrity cannot be compromised by an attacker who has full control of all network links, assuming no compromise of keying materials, installed software, or cryptographic algorithms.
medium	Command authenticity or integrity could be compromised by a packet substitution or DNS spoofing attack.

weak	Command could be forged by an attacker who has previously been a passive listener on one or more network links.
none	Command could be forged by an attacker with no special information.

Examples of medium protection include one-time passwords [RFC 2289] and HTTP digest authentication [RFC 2617 section 3]. Examples of weak protection include cleartext passwords or security protocols subject to replay attacks.

If a server or USER AGENT receives a command with no Astrength header, it should assume that the equivalent Astrength is "none".

A server relaying a command MUST communicate the weaker of the strength of the connection it received the command on and the Astrength value communicated from the last entity.

A server MAY choose to reject a command with a "410 AStrength Too Weak" error because it does not come with sufficient authentication strength (either as reported by the Astrength value or based on the connection from the immediate requester). A relay MUST NOT reject a response on the basis of insufficient authentication strength.

Note that, separately from connection-level authentication, an operation may be authenticated using an end-to-end signature. The Astrength header does not bear any relation to this kind of authentication.

An example scenario: a PRIM USER AGENT connects to a server for example.net and authenticates using a weak mechanism. It then issues a "send" command from alice@example.net to bob@domain.com. The example.net server connects to domain.com, authenticates using DNSSEC- signed public keys and forwards the IM with "Astrength: weak" because the previous link was authenticated with a weak. The domain.com server sends the command to the clients receiving commands for bob@domain.com with "Astrength: weak" since that was the authentication value claimed by example.net, even though domain.com received the command over a strongly authenticated link.

Another example scenario: a PRIM client connects to a server for example.net and authenticates using some strong SASL mechanism as alice. It then issues a "send" command from alice@example.net to bob@domain.com. The example.net server connects to domain.com and authenticates, but example.net's public key DNS record is not signed, so it could have been forged by a DNS spoofing attack. The example.net server sends the IM with "Astrength: strong" because it

received the command from Alice over a strongly authenticated link; however, the domain.com server will weaken the Astrength to "Medium" when forwarding the command to Bob's clients.

6.1.9. Date

Specifies the date and time this command was originally issued. PRIM adopts the date syntax as defined in Section 15.5, i.e. specified in [RFC1123].

```
date-header = "Date: " date-time
                        ; as defined in Section 15.5
```

[It will be affected by the CPIM specification because it would be preferable to have the same format with it. Need more discussions.]

6.2. Entity Headers

6.2.1. Content-Type

A command-body MUST NOT be included unless the description of the particular method allows it. If a command-body is included, the protocol command headers MUST include a Content-Type as specified in [RFC 2045].

The Content-Transfer-Encoding header from [RFC 2045] is not necessary and MUST NOT be included in any command or response. An implementation which receives a Content-Transfer-Encoding header should reject the command with an error 400 Bad Request.

6.3. Presence Headers

6.3.1. Duration

When used with a SUBSCRIBE command and its response, it specifies the amount of seconds the caused subscription SHOULD remain in effect for.

```
duration-header = "Duration: " 1*DIGIT
```

6.3.2. Subscription-ID

Specifies the unique identifier of the subscription in the watcher and presentity pair. This header MUST appear in the SUBSCRIBE and NOTIFY commands, and in the responses to SUBSCRIBE commands.

subscription-id-header = "Subscription-ID: " 1*(unreserved / escaped)

6.4. IM Headers

6.4.1. Message-ID

The message-id-header specifies the identifier of each IM, which distinguishes the message from others. The sender must generate a globally unique message-id for each IM sent.

message-id-header = "Message-ID" 1*(DIGIT / ALPHA) ": " im-id

6.4.2. Conversation-ID

The conversation-id is used in the SEND command to identify the conversation channel shared by the participants of an IM exchange. A "conversation channel" means a virtual channel which consists of a thread of IMs. When a PRINCIPAL replies to an IM, the reply MUST have the same conversation-id header.

conversation-id-header = "Conversation-ID: " 1*(unreserved / escaped)

6.4.3. Reply-To

The reply-to-header is optionally specified in a SEND command. It indicates an INSTANT INBOX identifier where the sender would prefer to receive any replies. The recipient SHOULD use the "reply-to" header, instead of the "from" header, if the former exists.

reply-to-header = "Reply-To: " im-id

7. Command Specifications

This section describes the command specifications for the PRIM commands used on the server-server connections. The commands are those for establishing connections, and exchanging PRESENCE INFORMATION and INSTANT MESSAGES.

In header descriptions below, the sign (o) on the right hand of a header indicates that the header is optional.

7.1. Presence Service Commands

This section describes the details of the protocol for the PRESENCE SERVICE.

7.1.1. SUBSCRIBE - Placement and renewal of SUBSCRIPTION

Headers:	Request	Response
	from-header	from-header
	to-header	to-header
	duration-header	duration-header
	subscription-id-header	subscription-id-header
	date-header	
	astrength-header (o)	
	content-type-header (o)	

The SUBSCRIBE method is used to express a WATCHER's interest on the PRESENCE INFORMATION of a PRESENTITY. There are two scenarios where the method is issued: when a

- WATCHER wishes to establish a new SUBSCRIPTION to a PRESENTITY, or
- Presence Server or USER AGENT needs to renew a SUBSCRIPTION on behalf of a WATCHER

from-header: identifies the WATCHER requesting the SUBSCRIPTION.

to-header: specifies the PRESENTITY to subscribe to.

duration-header: specifies the amount of seconds that this subscription is valid for.

subscription-id-header: specifies the unique identifier of the subscription within the watcher (requester) and the presentity.

date-header: specifies date and time when the command is generated.

astrength-header: specifies the authentication strength of the previous hops as described in Section 6.1.8.

The SUBSCRIBE command MAY have a command-body in order to present a piece of information to the target presence server. The meaning of the command-body depends on the services or implementations.

A response to the SUBSCRIBE command contains no command-body. After a successful (200 or 201) response to the command, a NOTIFY command which carries the presence information of the target PRESENTITY MUST be immediately invoked.

If the value of the duration-header of a SUBSCRIBE command is zero,

no subscription is established. In this case, if the subscription-id-header's value is the same one as an existing SUBSCRIPTION of the WATCHER to the PRESENTITY, that SUBSCRIPTION MUST be removed. If the value of the subscription-id-header does not match any of the existing SUBSCRIPTIONS of the WATCHER to the PRESENTITY, it has no impact on these SUBSCRIPTIONS and the SUBSCRIBE command behaves like "fetching" the PRESENCE INFORMATION.

The Return Codes are:

200 OK: The SUBSCRIPTION was placed successfully. The command contains no command-body.

201 Duration Adjusted: The SUBSCRIPTION was placed successfully, yet a different duration was set and this is indicated in the duration- header of the response.

402 Forbidden: The PRESENTITY authenticated in the current connection does not have rights (through the current ACL) to SUBSCRIBE to the PRESENTITY requested. No command-body is present.

403 Resource Not Found: The PRESENTITY does not exist. No command-body is present.

404 Subscription Not Found: The subscription specified by the subscription-id-header does not exist in the case of renewing SUBSCRIBE requests. No command-body is present.

505 Too Many Subscriptions: The maximum amount of SUBSCRIPTIONS placed by the system administrator or by the targeted PRESENTITY has been reached. No command-body is present.

If a SUBSCRIPTION already exists between a WATCHER and a PRESENTITY, then a successful SUBSCRIBE request from the WATCHER updates the duration of the SUBSCRIPTION to the value carried in the request.

7.1.2. UNSUBSCRIBE - Removal of SUBSCRIPTION

Headers:	Request	Response
	from-header	from-header
	to-header	to-header
	astrength-header (o)	

The UNSUBSCRIBE method indicates that the WATCHER is no longer

interested in receiving NOTIFICATIONS for changes in PRESENCE INFORMATION of a PRESENTITY.

It may either be issued by a USER AGENT or a Presence Server on behalf of the WATCHER.

The from-header identifies the WATCHER requesting the SUBSCRIPTION cancellation.

The to-header specifies the PRESENTITY to unsubscribe from.

The Response MUST NOT carry a command-body. The Return Codes in the Response are:

200 OK: The SUBSCRIPTION was removed.

404 Subscription Not Found: there is no SUBSCRIPTION from the specified WATCHER to the specified PRESENTITY.

Note: When the duration of a SUBSCRIPTION elapses, without the reception of a renewal, the Presence Server MUST assume an implicit UNSUBSCRIBE method has been received.

7.1.3. NOTIFY - Propagation of PRESENCE INFORMATION

Headers:	Request	Response
	from-header	from-header
	to-header	to-header
	subscription-id-header	
	astrength-header (o)	
	duration-header (o)	
	date-header	
	content-type-header	

The NOTIFY command informs WATCHERS when the PRESENCE INFORMATION of the PRESENTITY they have SUBSCRIPTIONS to has changed. Also, it MUST be issued immediately after processing of successful SUBSCRIBE commands. The NOTIFY will carry the whole PRESENCE INFORMATION and not just the modified tuple.

The command-body carries a presence document corresponding to the PRESENCE INFORMATION for the PRESENTITY. The body MUST be a data of the Content-Type "message/cpim" [CPIM-MSGFMT] containing an "application/cpim-pidf+xml" data [CPIM-PIDF], or just a data of the "application/cpim-pidf+xml" type.

The headers for this command are:

from-header: identifies the PRESENTITY this NOTIFICATION is about.

to-header: specifies the WATCHER that needs to receive this information.

astrength-header: specifies the authentication strength of the previous hops as described in Section 6.1.8.

duration-header: specifies the amount of remaining seconds that the corresponding subscription is valid for. Optional.

date-header: specifies date and time when the command is generated.

content-type-header: specifies the content type of the command-body.

The Response MUST NOT carry any command-body.

The NOTIFY command MAY contain the duration-header that specifies the amount of remaining seconds of the corresponding SUBSCRIPTION. If the value of the duration-header is zero, the NOTIFY command informs the WATCHER that the SUBSCRIPTION has been canceled by some means. No future NOTIFICATIONS will be sent to this WATCHER.

The Return Codes are:

200 OK: the PRESENCE INFORMATION was received and processed correctly.

400 Bad Request: The command was malformed or the command-body did not carry a valid XML document. The PRESENCE INFORMATION was not accepted.

403 Resource Not Found: no such WATCHER exists.

7.2. Instant Messaging Service Commands

This section describes the details of the commands for the INSTANT MESSAGE SERVICE.

7.2.1. SEND - Sending Messages

Headers:	Request	Response

from-header	from-header
to-header	to-header
message-id-header	message-id-header
conversation-id-header	conversation-id-header
date-header	
astrength-header (o)	
content-type-header	

[Note. It would be necessary to make the "SEND" command syntax compatible with the CPIM specification. We need more discussion.]

The SEND command is used to transport an INSTANT MESSAGE.

The command-body carries an INSTANT MESSAGE, as described in section 11.

The headers for this command are:

from-header: identifies the SENDER of the message.

to-header: identifies the INSTANT INBOX the message is sent to.

astrength-header: indicates the lowest authentication strength for previous hops of the command.

message-id-header: specifies a unique identifier for each INSTANT MESSAGE.

conversation-id-header: specifies a unique identifier to distinguish a given conversation thread between multiple participants.

date-header: specifies date and time when the command is generated.

content-type-header: specifies the content type of the command-body.

The response to this method MUST NOT carry any command-body, and MAY have the following return codes:

101 Unknown Delivery Status: The IM Service cannot assure that the message was delivered.

200 OK: the INSTANT MESSAGE was delivered at least to one PRINCIPAL that was listening to the recipient INSTANT INBOX.

402 Forbidden: The PRINCIPAL authenticated in the current

connection does not have rights (through the current ACL) to send messages to the recipient INSTANT INBOX.

408 Inbox Is Closed: the INSTANT MESSAGE could not be delivered because the recipient INSTANT INBOX was closed. This may be issued by either the IM Server if there is no-one listening to that inbox, or by a USER AGENT if it decides to block the sender (see explanation below).

The response code sent by the IM Server hosting the recipient INSTANT INBOX is always the most positive response from all the connections listening to that INBOX. Thus, if at least one USER AGENT acknowledges the message, then its server will acknowledge it too.

Note: It is important to remember that PRESENCE INFORMATION may be revealed through the responses to INSTANT MESSAGES. For example, it may be possible for someone to "ping" an INSTANT INBOX by sending messages to it, in order to deduce PRESENCE INFORMATION from the state of that INBOX. USER AGENT implementations can prevent that if necessary by returning 408 Inbox Is Closed if the sender of an INSTANT MESSAGE should not know that the INBOX is OPEN.

7.3. General Commands

The commands described in this section apply to both the PRESENCE and INSTANT MESSAGING services.

7.3.1. LOGIN - Connection Setup

Headers:	Request	Response
	domain-header	domain-header
	auth-state-header	auth-state-header
	SASL-mechanism-header	SASL-mechanism-header

For a server-server connection, the initiating host MAY issue a LOGIN request prior to sending any presence or IM commands in order to authenticate itself as an authoritative host in the initiating domain. The domain MUST be presented with the domain-header.

If the authentication process is not successful the TCP connection MUST be dropped. The LOGIN request MAY be preceded by the STARTTLS request when the implementations support TLS for a secure connection. Any other requests that are received before the authentication completed MUST receive an "Unauthorized" response.

The authentication process is not necessarily completed in a single request/response pair, but it can be fulfilled in a sequence of the request/response pairs. The auth-state-header MUST be used to indicate the state of the authentication process.

The command-body in the LOGIN request and its response MAY carry the authentication information for the respective SASL mechanism.

See section 9 for the details of authentication procedures.

Return Codes:

100 Authentication Continued: This response may possibly carry a command-body with information pertaining to the SASL challenge, and a SASL-mechanism-header specifying the SASL mechanism supported by the server. The originator needs to send other LOGIN command, with auth-state-header as "continue", and the response to the challenge in the command-body.

200 OK: The sender is authenticated and the connection may be used to transport further commands.

406 Authorization Failed: The operation failed to authenticate the connection. No further commands are allowed and the receiver MUST terminate the connection.

409 Already Authenticated: This is returned if a LOGIN command has already succeeded.

7.3.2. STARTTLS - Secuire Connection Setup

Headers:	Request	Response
	none	none

A server MAY issue STARTTLS request to upgrade a TCP connection to a TLS [TLS] enabled one. Implementations that support TLS MAY issue a STARTTLS request prior to issuing any other requests.

Once the client credentials are successfully exchanged using TLS negotiation, the "EXTERNAL" SASL mechanism MAY be used in the subsequent LOGIN process. The "PLAIN" SASL mechanism SHOULD NOT be used if the STARTTLS upgrading process fails to establish a fully strong encryption layer.

The Request and the response MUST NOT carry a command-body.

Return Codes:

200 OK: The TLS negotiation should start. Once a STARTTLS command issued, the initiator MUST NOT issue further requests until a server response is received and the TLS negotiation is completed.

501 Not Implemented: TLS is not implemented and thus the client must authenticate itself using the LOGIN method.

7.3.3. LOGOUT - Connection Shutdown

Headers:	Request
	none

The receiver of the LOGOUT command MUST NOT send any response.

7.3.4. PING - Testing a connectionG

Headers:	Request
	none

When a peer in a connection wants to verify if the connection is alive, it may send a PING command. No response is expected from the other peer.

A successful transmission of a PING does not guarantee its reception at the other end, nor does it verify that all is well with its peer. However the transmission of the PING may provoke an error, and thereby causing the sending peer to realize there is a problem with the connection. If this happens the USER AGENT or server assumes an implicit LOGOUT command.

7.3.5. VERIFYSERVER - Verifying a server's authority

Headers:	Request	Response
	server-address-header	server-address-header

As described in section 13.2, when a server needs to verify whether another server (known through its IP address) belongs to a given domain, it performs one or more DNS lookups. Large domains with a significant amount of servers might not be able to publish every

entry for every server, due to DNS limitations. Thus a DNS lookup might not be sufficient to determine whether a given server belongs to a given domain.

If it is not possible to verify the domain of a server through a DNS lookup, a VERIFYSERVER command can be issued.

The VERIFYSERVER MAY be issued in a new TCP connection, without previous LOGIN. The verifying server will issue the command to any of the addresses returned in the DNS lookup.

The server-address-header specifies the IP address of the server that needs verification.

The response MUST NOT have a command body.

Return Codes:

200 OK: the server does belong to that domain.

403 Resource Not Found: the server does not belong to this domain.

8. Response Codes

The policy for assigning response codes follows the convention used in HTTP/1.1 [HTTP1.1].

- o 1xx: Informational - Request received, continuing process
- o 2xx: Success - The action was successfully received, understood, and accepted
- o 3xx: Redirection - Further action must be taken in order to complete the request
- o 4xx: Request Error - The request contains bad syntax or cannot be fulfilled
- o 5xx: Server Error - The server failed to fulfill an apparently valid request

100 Authentication Continued

The request for authentication has been accepted and the authentication process is continued.

101 Unknown Delivery Status

The server was unable to determine that the message was successfully delivered to an INSTANT INBOX or that the transmission failed. This could be because the message was delivered on a best-effort basis, or it was delivered to an "offline" message store.

200 OK

The request has been successfully processed.

201 Duration Adjusted

The SUBSCRIPTION was placed successfully, yet its duration was not acceptable to the server. A new duration was set and this was indicated in the duration-header of the response.

300 Redirect

The server was unable to deal with the request and instructs the caller to reconnect to a different server and reissue the operation there.

400 Bad Request

The request could not be understood by the server due to malformed syntax of the headers or malformed content. The requesting host SHOULD NOT repeat the request without modifications.

401 Unauthorized

The request requires authentication. If received this response, the requesting host MUST authenticate itself through the LOGIN request.

402 Forbidden

The server understood the request, but it has not been authorized.

403 Resource Not Found

The specified resource was not found at the server.

404 Subscription Not Found

The SUBSCRIPTION specified in the Subscription-ID header was not found at the resource.

406 Authentication Failed

The authentication process has failed. The reason for it is one of the following:

- o The authentication process using the specified SASL-Mechanism failed.
- o The LOGIN request specifies an inappropriate SASL Mechanism.
- o In the midst of the authentication process, the requester tries to start another authentication process by specifying 'Auth-State: init'.

407 Timeout

The server timed-out after waiting for a response from another client or server.

408 Inbox Is Closed

The INSTANT INBOX is not currently accepting messages.

409 Already Authenticated

The connection was authenticated previously through a LOGIN command.

410 Astrength Too Weak

The command was rejected because the server requires a higher level of security and this could not be provided.

500 Internal Server Error

The request has not been fulfilled because of the error internal to the server.

501 Not Implemented

The server does not support the functionality required to fulfill the request. This response is typically returned when the server has received a request of the services it does not provide.

502 Bad Gateway

The server, while acting as a gateway or proxy, received an invalid response from the other PRIM or non-PRIM server it accessed in attempting to fulfill the request.

503 Version Not Supported

The server or client does not support the specified protocol version used for the request.

504 Gateway Timeout

The server, while acting as a gateway or proxy, did not receive a timely response from the other PRIM or non-PRIM server it accessed in attempting to fulfill the request.

505 Too Many Subscriptions

The SUBSCRIBE request has not been fulfilled because the request exceeds the specified maximum number of SUBSCRIPTIONS at the resource. When this status code is received, the client SHOULD NOT retry the SUBSCRIPTION immediately.

9. Authentication

PRIM implements security on a per-connection basis: each connection end-point is authenticated in order to establish the PRESENTITIES and INBOXES on behalf of which that end-point can communicate.

After authentication succeeds, the other end-point will accept requests pertaining to those PRESENTITIES or INBOXES, and direct requests to them over that connection.

9.1. Server-Server Authentication

When a server establishes a connection to another server, that connection end-point can be authorized to communicate on behalf of multiple PRESENTITIES or INBOXES. This is usually performed by authenticating the end-point as a valid host of the initiating domain.

The connection end-points MAY use a LOGIN command for the authentication. But, if two end-points have authenticated each other with an out-of-band method (e.g. TLS) or they have enough knowledge about the other end-point for their purpose (e.g. IP address), it MAY NOT use such a particular command. If the connection uses TLS, then the domains served by each end-point are established in the beginning, through the certificates provided.

Another type of authorization can take place throughout the duration of the connection. Each end-point will establish that a domain is being served by the other end-point when the first request pertaining

to that domain is received. This can happen more than once per connection.

Verification that a given server is responsible for a given domain is done by performing a name resolution (as described in section 7.2). It is possible that due to DNS limitations, in the case of a domain with a large number of servers, only partial DNS records are advertised. Thus, the address of the server initiating the connection may not be in the records received. In this case a VERIFYSERVER method is performed to establish whether the initiating server has authority over the corresponding domain. This is described in Section 10.5.

9.2. Authentication Using LOGIN

When a server establishes a connection to a server, it MAY issue a LOGIN command to authenticate itself as a valid host representing its domain. The LOGIN authentication procedure in PRIM uses SASL [SASL]. The LOGIN request and response MUST include a SASL-Mechanism header field so that the end-points could negotiate the SASL mechanism to be used. As SASL mechanisms, every PRIM implementation MUST implement PLAIN [SASL-PLAIN] and is RECOMMENDED to implement CRAM-MD5 [CRAM-MD5], and EXTERNAL [SASL]. It MAY also implement other mechanisms as needed.

The LOGIN authentication procedure is sketched as follows;

(1) The initial LOGIN request

A server issues a LOGIN request including the Auth-State header with the value "init". It MUST also contain the Domain header specifying the initiating domain and the SASL-Mechanism header whose value is a comma-separated list of SASL mechanisms the initiating host is capable to use in the descending order of preference.

[Example]

```
LOGIN PRIM/1.0 0224 0
Domain: prim.fujitsu.com
Auth-State: init
SASL-Mech: CRAM-MD5,PLAIN
```

If the LOGIN request is acceptable, the target server SHOULD respond with '100 Authentication Continued' response. It MUST contains the SASL-Mechanism header with the value of at least one

selected SASL mechanism by the server. If a challenge-response mechanism is selected, the response MUST contain a challenge data in the body.

```
PRIM/1.0 0224 48 100 Authentication Continued
Domain: prim.fujitsu.com
Auth-State: init
SASL-Mech: CRAM-MD5

<20010226095208.1018677043.foo1.bar.fujitsu.com>
```

(2) The subsequent LOGIN requests

If an initiating server receives a '100 Authentication Continued' response to the initial LOGIN request, it SHOULD try another LOGIN request with the header 'Auth-State: continue'. This LOGIN request MUST contain the SASL-Mechanism header with the single value of selected SASL mechanism.

The LOGIN request MAY contain the domain's authentication information in the body required by the selected mechanism. Details in the case of CRAM-MD5, PLAIN, and EXTERNAL are described in the following subsections.

If the LOGIN request is validated, the target server respond with a '200 OK' response. If the same PRINCIPAL is already authenticated by a preceding LOGIN procedure, the server MAY respond with a '409 Already Authenticated'. Otherwise, a '406 Authentication Failed' response SHOULD be returned to the USER AGENT. In this case, the USER AGENT MUST NOT send any other request commands in this connection.

(2-a) CRAM-MD5

The USER AGENT calculates the response data using the keyed MD5 algorithm [KEYED-MD5] where the key is the shared pass phrase and the text is the challenge data. Then, it sends the hexadecimal string of the response octets prepended by the user name and CRLF in the body of the LOGIN request.

[Example]

```
LOGIN PRIM/1.0 0225 50
Domain: prim.fujitsu.com
Auth-State: continue
SASL-Mech: CRAM-MD5
Content-Type: text/plain
```

```
prim.fujitsu.com
106d12b16fc323dc2f3d19b587f8d0ff
```

(2-b) PLAIN

The PLAIN mechanism SHOULD be used only on a fully secured connection, such as one already encrypted with TLS. In this case, the body part of the LOGIN request contains the user name and the pass phrase in a plain text separated by CRLF.

[Example]

```
LOGIN PRIM/1.0 84505230 27
Domain: prim.fujitsu.com
Auth-State: continue
SASL-Mech: PLAIN
Content-Type: text/plain

prim.fujitsu.com
hi there!
```

(2-c) EXTERNAL

The EXTERNAL mechanism is intended to be used in the case that the PRINCIPAL has been already authenticated with some external authentication method, such as TLS mutual authentication. The LOGIN command using this mechanism contains nothing in the body.

10. Presence Information Data Format (PIDF)

PRIM adopts CPIM Presence Information Data Format [CPIM-PIDF] as its presence data format. This brings CPIM conformance to PRIM with its native presence data format. The content-type "application/cpim-pidf+xml" is defined in that specification.

See the reference for detailed information.

11. IM Format

INSTANT MESSAGES are opaque payloads transferred by SEND commands tagged by a MIME [MIME] content type.

A SEND command MUST contain a Content-Type header which specifies

the content type of the payload. It MAY contain any proper MIME header which may not be defined here.

For the CPIM conformance, A USER AGENT MUST understand and generate messages of the content type 'message/cpim'[CPIM-MSG]. In particular, a USER AGENT MUST generate an INSTANT MESSAGE of the type 'message/cpim' if it sends the message to other domains which do not or may not understand PRIM. The correspondence between the PRIM and CPIM message format is described in Section 17.

The PRIM servers MUST forward the message as is when the message is relayed to the clients or other servers. That is, the servers MUST NOT delete or modify any header which appears in the command.

12. Security Considerations

There exists many kind of security threats in the Presence / Instant Messaging services and applications as described in the IMPP Requirements [RFC 1778] and the CPIM document [CPIM].

PRIM specifies mechanisms to achieve connection security and to realize access control including presence publication control.

The future PRIM specifications will conform to the expected CPIM data formats for secure and interoperable Presence information and IM exchanges [CPIM,CPIM-MSG]. It will acquire the message level security such as end-to-end confidentiality and integrity.

13. References

[CPIM] D. Crocker et al., "A Common Profile for Instant Messaging (CPIM)", draft-ietf-impp-cpim-01.txt, Work in Progress.

[CPIM-MSG] D. Atkins and G. Klyne, "Common Presence and Instant Messaging Message Format", draft-ietf-impp-cpim-msgfmt-00.txt, Work in Progress.

[CRAM-MD5] J.Klensin, R.Catoe and P. Krumviede, "IMAP/POP AUTHorize Extension for Simple Challenge/Response", RFC 2195, September 997.

[HTTP1.1] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999

[MIME] Multipurpose Internet Mail Extensions. See RFC 822, RFC 2045, RFC 2046, RFC 2047, RFC 2048, and RFC 2049.

[OpenPGP] J. Callas, etc., "OpenPGP Message Format", RFC2440, November 1998.

[RFC822] Crocker, D., "Standard for the format of ARPA Internet text messages", RFC 822, STD 11, Aug 1982.

[RFC1123] R. Braden, "Requirements for Internet Hosts -- Application and Support", RFC 1123, October 1989

[RFC1738] T. Berners-Lee, L. Masinter, M. McCahill, "Uniform Resource Locators", RFC 1738, December 1994.

[RFC2778] M. Day, J. Rosenberg, H. Sugano, "A Model for Presence and Instant Messaging", RFC 2778, February 2000.

[RFC2779] M.Day, S.Aggarwal, G.Mohr, and J.Vincent, "Instant Messaging / Presence Protocol Requirements", RFC 2779, February 2000.

[SASL] J. Myers, "Simple Authentication and Security Layer (SASL)", RFC2222, October 1997.

[SASL-PLAIN] C. Newman, "Using TLS with IMAP, POP3 and ACAP", RFC2595, June 1999.

[SMIME] P. Hoffman, Ed, "S/MIME Version 3 Message Specification", RFC2633, June 1999.

[TLS] T.Dierks, and C. Allen, "The TLS Protocol Version 1.0", RFC2246, January 1999.

[URI] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC2396, August 1998.

[XML] Extensible Mark Up Language. A W3C recommendation. See http://www.w3.org/TR/1998/REC-xml-19980210 for the 10 February 1998 version.

14. Acknowledgements

The authors greatly appreciate helpful comments from John Stracke and Harald Alvestrand.

15. Author's Addresses

F. Mazzoldi
flo@personity.com
Personity, Inc.
4516 Henry Street, Suite 113
Pittsburgh PA 15213
USA

A. Diacakis
thanos@personity.com
Personity, Inc.
4516 Henry Street, Suite 113
Pittsburgh, PA 15213
USA

S. Fujimoto
shingo_fujimoto@jp.fujitsu.com
Fujitsu Laboratories Ltd.
64, Nishiwaki
Ohkubo-cho
Akashi 674
Japan

G. Hudson
ghudson@mit.edu
Massachusetts Institue of Technology
Cambridge, MA 02139
USA

J. D. Ramsdell
ramsdell@mitre.org
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730-1420
USA

H. Sugano
sugano.h@jp.fujitsu.com
Fujitsu Laboratories Ltd.
64, Nishiwaki
Ohkubo-cho
Akashi 674
Japan

16. Full Copyright Statement

Acknowledgement

Funding for the RFC editor function is currently provided by the Internet Society.