INTERNET-DRAFT
Expires: December, 2000

H. Sugano
A. Iwakawa
K. Otani
T. Ohno
S. Fujimoto
Fujitsu
June 2000

Privacy-enhanced Presence Protocol (PePP)
<draft-sugano-impp-proposal-pepp-00.txt>

Status of this Memo

Copyright Notice

Abstract

This document describes a protocol designed for scalable and secure Instant Messaging and Presence Services. The protocol, Privacy enhanced Presence Protocol (PePP), has been developed for an experimental Presence Service and recently extended to satisfy a variety of requirements for the Internet-wide, interoperable standards. This is a protocol proposal for the IMPP Working Group.

Table of Contents

1. Introduction 4
2. Terminology 4
3. Protocol Overview 6
3.1. Architecture 6
3.2. Model for Presence Service 7
3.2.1. Privacy Requirements 7
3.2.2. Presence Sections 7
3.2.3. Section based Access Control 8
3.2.4. Lease Model of Presence Information 9
3.2.5. Two Modes of Subscription 10
3.3. PePP Connections 10
3.3.1. Client Connections 11
3.3.2. Server Connections 13
3.3.3. Direct Connections 15
3.4. Subscription Model 16
3.4.1. Behavior of Clients 17
3.4.2. Behavior of Home Servers 17
3.4.3. Behavior of Target Servers 18
3.5. Instant Messaging Service 18
3.5.1. Basic Architecture 18
3.5.2. IM Conversation 19
3.5.3. Multiparty Conversation 19
3.6. Character and Content Encoding 20
4. Considerations Regarding IMPP Requirements 20
4.1. Scalability 21
4.2. Security 21
4.3. Wireless 23
4.4. Separability of Services 23
5. PePP Messages 24
5.1. Message Overview 24
5.2. PePP Addresses 25
5.3. Message Syntax 25
6. PePP Headers 27
6.1. From 27
6.2. Connection-Mode 27
6.3. Max-Content-Length 27
6.4. Subscription-Mode 28
6.5. Subscription-ID 28
6.6. Regarding 28
6.7. Change-Mode 29
6.8. Cancel-Type 29
6.9. Duration 30
6.10. Last-Modified 30
6.11. Section-ID 31
6.12. Section-Name 31
6.13. Location 31

6.14. Content-Type 32
6.15. Content-Length 32
6.16. Auth-State 32
6.17. SASL-Mechanism 32
6.18. Message-ID 32
6.19. Conversation-ID 33
7. PePP Methods 33
7.1. LOGIN 33
7.2. LOGOUT 34
7.3. SUBSCRIBE 35
7.4. UNSUBSCRIBE 36
7.5. REQUESTNOTIFY 37
7.6. CHANGE 38
7.7. CANCEL 39
7.8. FETCH 40
7.9. NOTIFY 41
7.10. PULL 42
7.11. SEND 42
7.12. RECEIVE 43
7.13. CALLBACK 44
7.14. REDIRECT 45
7.15. SETACL 45
7.16. GETACL 46
7.17. CREATESECTION 46
7.18. DELETESECTION 47
7.19. PING 47
7.20. STARTTLS 48
7.21. CONNECT 49
8. Status Codes 49
8.1. 1xx 50
8.2. 2xx 50
8.3. 3xx 50
8.4. 4xx 51
8.5. 5xx 52
9. Presence Information Data Format 53
9.1. Overview 53
9.2. Tag Descriptions 54
10. Subscribers Information 57
11. Access Control List 58
11.1. Overview 58
11.2. ACL Functions 58
11.3. Syntax of ACL 59
12. Sample Transcripts 61
13. Security Considerations 65
14. Acknowledgments 65
15. References 65
16. Authors' Addresses 66

1. Introduction

Instant Messaging (IM) and Presence Information (PI) Services have received broad attention as an emerging technology for real-time communication on the Internet. While there are a couple of services already deployed and widely used, each of those is based on its proprietary protocol and users cannot make use of IM Services like e-mails. This is a serious problem to overcome from both the technical and industrial standpoints. To solve the problem, the Instant Messaging and Presence Protocol (IMPP) WG has been formed at IETF, and been working to create an open, interoperable standards for IM/Presence technologies.

We at Fujitsu have long been interested in the development of the Internet-wide standards for IM and Presence services. To this end, we have collaborated with other players and contributed to the design of the standards. At the same time, we have been working on an Instant Messaging/Presence protocol called PePP for experimental client and server development. PePP is still a work in progress, and the current implementation is mainly concerned with the single domain utilization. Thus, we have been working to improve PePP to make it satisfy the IMPP requirements [Reqts] based on our interest for the interoperable standard.

Our main concerns in the development of PePP are security/privacy and scalability. In particular, as the Presence Service is considered to be fairly privacy sensitive, PePP is designed to have a means for fine privacy control on publishing a variety of Presence Information. For scalability, PePP has some features to avoid the server and connection bottlenecks.

This document describes the present version of the extended PePP specification. The authors submits this document as a proposal for the IMPP standardization. Further, we wish to share our ideas in the PePP design with the community to spur further discussion of the development of the standard. We would welcome any comments, suggestions and evaluations on PePP.

2. Terminology

This document makes use of the vocabulary defined in the IMPP Model and Requirements documents [Model,Reqts]. The capitalized terms such as PRESENTITY, WATCHER, PRESENCE SERVICE are used in the same meaning as defined in [Model,Reqts] unless otherwise stated. Some newly introduced terms are also defined here.

A "PePP Server", or just a server is a logical entity which provides the PePP IM/Presence services. A "PePP Client", or just a client is a logical entity which exchanges IMs and/or Presence Information interacting with the servers. Note that, although the IMPP Model document defines several types of ideal clients such as PRESENTITY, WATCHER, SENDER, and INBOX, the PePP client is an entity which may integrate these functions.

A "Domain" in the context of PePP is an administrative entity of the IM/Presence services. A user of the IM/Presence services in PePP has an account in a domain, and the user can get the services using one or more clients to connect to the servers in the domain. We call the domain in which a user has an account the "Home Domain" of the user.

For a user or the user's client, the "Home Server" is a server in the user's home domain which maintains and publishes Presence Information of the user. As stated in Section 3, the client only has a direct connection with the Home Server, and the user controls her/his Presence Information using the client.

A "Resource" in the context of PePP is a data unit in the PePP server, at which Presence Information of a particular PRESENTITY is published so that WATCHERS could access it. Thus, a resource is a unit for controlling and publishing Presence Information. Each resource is managed by the user controlling the PRESENTITY whose Presence Information is published at the resource. The user is called the "Owner" of the resource.

A "PePP Address" is an identifier to locate the PePP resource, which is represented by a URI. This is defined in Section 5.

If a client tries to access the Presence Information of another user, the user's Home Server is sometimes called the "Target Server" from the viewpoint of the client.

A "PePP message" or just a "Message" is the unit of PePP communication, consisting of a structured sequence of octets matching the syntax defined in Section 5. As defined there, a message is a "Request" message or a "Response" message. A "Message Body" is a part of a "Message" as defined in the same definition. Note that a "PePP message" has a different meaning from that of messages when we talk about "Instant Messages".

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] .

3. Protocol Overview

3.1. Architecture

PePP is designed for scalable and secure Instant Messaging and Presence (IM/P) Services. Because IM/P Services is usually provided as an integrated service, we have designed PePP as a single protocol for both services.

The PePP architecture involves two kinds of components; clients and servers. Clients may serve as user agents to the IM/P services, but may also be other software entity to utilize the services. Servers may or may not be different for IM and Presence services, but we assume in this document that a single "Home Server" provides both service for a user.

PePP adopts a Client-Server-Server-Client architecture. This means that a client only communicates with its home servers, and only servers can communicate with other servers which are possibly located in different domains. All messages exchanged between a client and a server and between a server and another server are transferred through TCP connections called "PePP connections". A PePP connection has two basic modes; the Client mode and Server mode. The Client mode is for a PePP connection between a client and a server, and the Server mode is for between servers. For descriptive simplicity, we use the terms "PePP Client Connections" and "PePP Server Connections" for PePP connections in the Client and Server mode, respectively. Fig.1 roughly depicts this architecture.

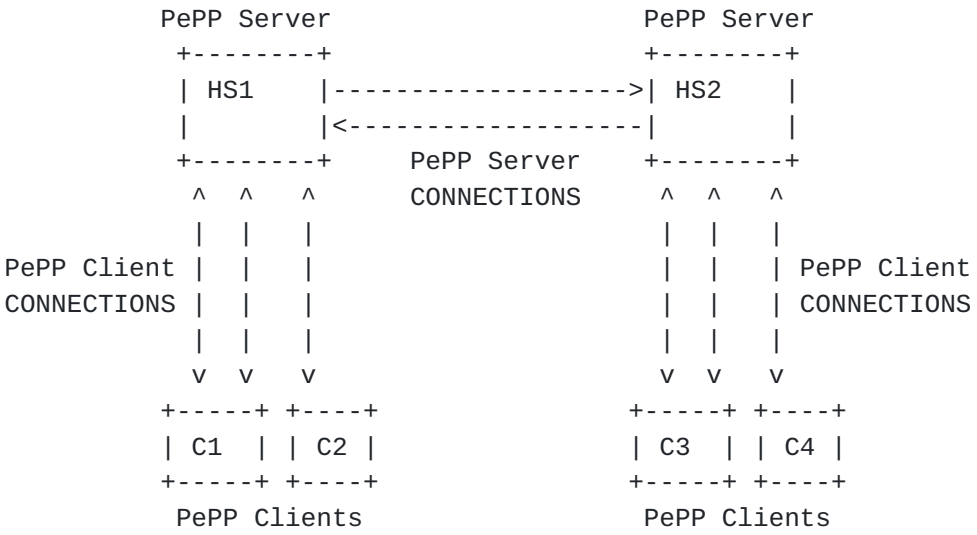

Fig.1: PePP Architecture

We assume that all the server/server relations could be fully trusted once they are authenticated each other based on appropriate authentication schemes. Thus, when a user in a different domain tries to access your Presence Information, you assume the user is already autheticated by her home domain and trust her identity. Furthermore, as we assume all the notification messages from other servers come through already established PePP Client Connections, the client can trust the authenticity of the notifications without extra authentication.

3.2. Model for Presence Service

PePP provides a means of fine privacy control of Presence Information publication.

3.2.1. Privacy Requirements

The IMPP Requirements document [Reqts] stipulates a variety of privacy requirements which IM/Presence services must meet. In addition to "confidentiality" as the most basic requirement, it states that a means for controlling privacy is necessary based on the observation that users are inclined to hide their activities from the public, and further they sometimes want to block a particular user from subscribing to their activity without letting the subscriber know their intention. This is a requirement for so-called "Polite Blocking" (5.1.15 [Reqts]).

Another requirement for the Presence Service is that users want to show themselves differently to different watchers (5.2.3 [Reqts]), which is considered as a variation on "Polite Blocking". We call this a requirement for "Personae".

These requirements lead our design practice to have multiple pieces of presence information that can be selectively shown to different subscribers.

3.2.2. Presence Sections

The PePP model considers Presence Information contained in a PePP resource as a collection of several pieces, each of which is called a "presence section". A presence section may contain a status information of a communication means, that of the user, or any other. A presence section can be considered as an embodiment of the notion of a PRESENCE TUPLE, which is defined by the Model document [Model].

The notion of presence sections is intended to be a unit for individual control of publication and filtering. For publication control, PePP allows the user controlling a PRESENTITY to set an access control list per section. For filtering control, PePP allows a WATCHER trying to subscribe to a PePP resource is capable of selecting presence sections to be notified their presence change.

Considering the privacy requirements such as "Polite Blocking" and "Personae", there may be a case such that the user controlling a PRESENTITY wants to hide completely which presence sections are shown to the WATCHERS. To realize this in PePP, each presence section has a "section name" as an identifier for the WATCHERS in addition to its unique identifier "section ID". The section ID is used to manipulate the content or control information of the presence section and it is not shown to WATCHERS. Instead, a section name is shown to WATCHERS.

An example of a structure of Presence Information (PI) in a PePP resource is shown in Fig.2. Here, an entire PI consists of several sections and each section contains a section ID, a section name, status, note, and an optional communication address. The syntax given here is temporary and the actual syntax of PI is defined in Section 9.

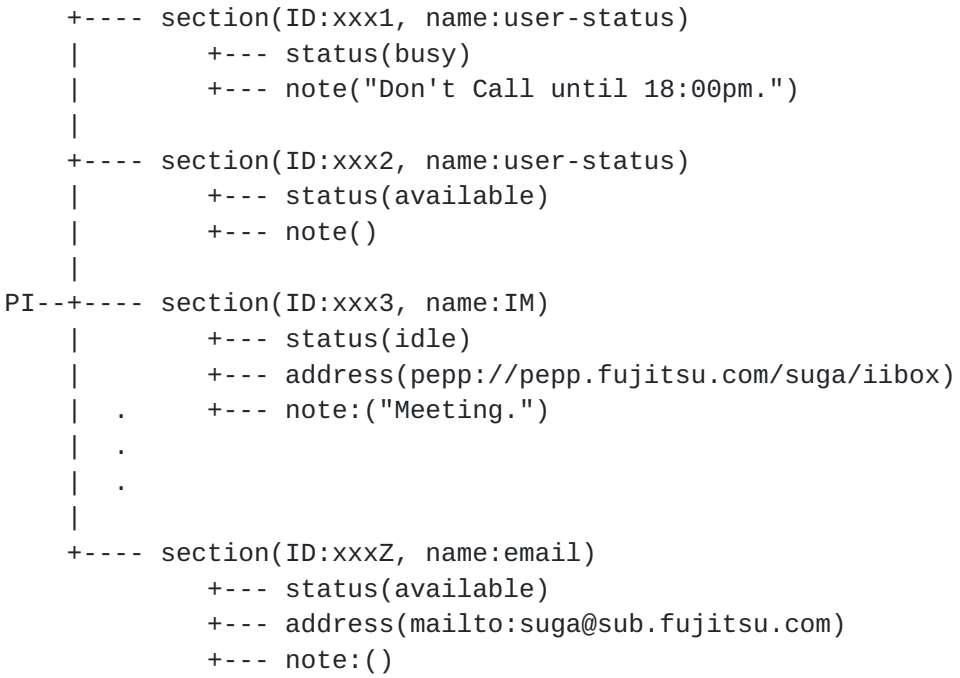

Fig.2: Presence Information as a set of Presence Sections

3.2.3. Section based Access Control

Access control in the Presence Service typically controls how to treat requests from WATCHERS. As stated above, PePP assumes the Presence Information consists of multiple presence sections, and each presence section can have different access control list (ACL).

When a request for subscription to a resource is received, the presence server evaluates the ACL of the resource to determine which sections should be shown to the WATCHER. Because several sections MAY have the same section name, this mechanism can be used for implementing a feature of "Personae".

In the example of Fig.2, the sections "xxx1" and "xxx2" have the same section name "user-status". Consider these sections have ACLs such that the section "xxx1" accepts user A but denies user B, and the section "xxx2" accepts user B but denies user A. When a user A and user B try to subscribe to this resource, user A receives the section "xxx1" as "user-status" and user B receives the section "xxx2" with the same name.

A WATCHER may receive several sections with the same name according to the ACCESS RULE, and we do not eliminate this situation in the protocol level. Individual implementation MAY select one of those sections with the same name.

Note that ACLs for the requests other than that for subscription is not set for a section, but for a resource.

3.2.4. Lease Model of Presence Information

PePP adopts the lease model for changing presence information. That is, a PRESENTITY MAY have two pieces of presence information, a lease value and a permanent value, for each section of the presence information. The lease value is associated with its duration value and the client renews the lease value within the duration to keep the lease. Otherwise, the value of the presence section turns back to the permanent value.

This feature is preferable because it provides a general solution to handle presence status or availability of various kind of communication means. While availability of some communication means such as IM is subject to unexpected failure or constantly changing communication environment, that of other communication means might always be acquirable from a particular entity. The latter does not have to use the lease value and just change the permanent value of the presence section. The lease model gives flexibility to the control of presence information.

3.2.5. Two Modes of Subscription

In the case a WATCHER subscribes to a resource publishing Presence Information of a PRESENTITY, the WATCHER usually requires a change notification when the Presence Information is modified. However, a WATCHER sometimes prefers to fetch the Presence Information rather than being notified every time. As PePP has a section based feature of controlling Presence Information, it also provides WATCHERS with a means to choose whether or not to receive notifications for each section.

In PePP, a subscriber can receive permitted presence sections within a single subscription to a resource. Moreover, within the subscription, the subscriber can restrict the sections to be notified change notifications. The selected sections are said to be in the Notify mode and the other ones are in the Pull mode. If all the sections are in the Notify mode, the subscription is the one in a usual sense and is called a Notify mode subscription. In a Pull mode, the client fetches the interested presence section when it wants. It is desirable when the subscriber wants to reduce the frequency of notification, for instance, in the case the user has a PePP client on a cellular phone device which charges per packet. This feature of PePP also realizes so called Selective Subscription.

The subscriber in the Pull mode subscription can be considered as a notion of FETCHER defined by the IMPP Model document [Model]. Therefore, the notion of subscribers in PePP coincides that of WATCHERS of the Model document, and the Subscribers Information (see Section 10) corresponds to the WATCHER INFORMATION in it. This is so designed because we believe that a WATCHER should be required to declare the interest on the PRESENTITY whether she/he wants to get notifications or not.

The Pull mode subscription may possibly cause a heavy load on the server. So, the server SHOULD be able to disallow it based on its policy.

3.3. PePP Connections

All PePP connections are TCP connections. We adopted TCP because it is widely used as a reliable transport on the Internet and we believe all the messages in PePP, not only IMs but also changes and notifications of Presence Information, must be safely transported. It is also favorable in the existence of firewalls because UDP datagrams are not usually permitted to come into through firewalls.

As stated above, PePP mandates two kinds of connections; PePP client

connections and PePP server connections. Moreover, as an OPTIONAL specification, we propose a mechanism to establish a virtually direct connection between clients for the sake of the end-to-end security.

3.3.1. Client Connections

A PePP client connection is a TCP connection between a client and its Home Server (HS). It is established only by the requests from clients. Clients can create more than one PePP connections if necessary. However, the first connection between the client and server plays a designated role as a "main" connection, and it is expected to be persistent during the service. Other connections are called "backup" connections.

(a) Establishing a Connection

On establishing a PePP client connection, a client opens a TCP connection to its HS and issues a LOGIN request message to the HS to start an authentication process. In the LOGIN request, the client MUST specify information about the authentication process (AUTH-STATE header), a connection mode (CONNECTION-MODE header), and MAY specify the maximum size of a message body it wishes to receive (MAX-CONTENT-LENGTH header). The client MUST specify "client" as the value of the Connection-Mode header field.

PePP connections use SASL [SASL] as an authentication framework. The client and server negotiate the SASL Mechanism to be used by the SASL-Mechanism header field. SASL Mechanisms supported in the PePP LOGIN process are CRAM-MD5 [CRAM-MD5], EXTERNAL [SASL], and PLAIN [SASL-PLAIN].

1) CRAM-MD5 - It can always be specified.
2) EXTERNAL - It uses TLS client authentication, and can be specified only if TLS client authentication is supported [TLS].
3) PLAIN - It can be specified only if TLS encryption is enabled.

The authentication process MAY be completed in a sequence of LOGIN requests and their responses. If the process terminated successfully, the last response MUST contain the fields specifying the connection ID (CONNECTION-ID header) and the MAX-CONTENT-LENGTH header field. The server MAY overwrite the value of the MAX-CONTENT-LENGTH specified by the client.

If a client wishes a secure transport, it MAY issue a STARTTLS request prior to the LOGIN request in order to upgrade the established TCP connection to a TLS enabled secure one. The TLS layer simply encrypts the whole PePP messages on the top of a TCP

connection, and provides secure communication channels between connection peers.

(b) PePP Messages

Once the PePP connection is established, it is used to send PePP request and response messages, which have similar syntax to HTTP messages, for the Presence and IM services. Like HTTP, a PePP request message generates a corresponding single response message. However, unlike HTTP, the PePP client connection can be used by both of the client and server to send the PePP request messages in both directions.

Because the "main" client connection is supposed to be persistent, either ends of the connection MUST send PING requests periodically in order to confirm the other is alive.

A PePP connection allows request pipelining, i.e. a client can send another request to the same connection before receiving the response to the previous request. Moreover, PePP allows the responses can be received in the different order from that of requests in order to avoid the server bottleneck caused by the processing overhead. To this end, every PePP messages MUST contain a Request-ID which is a unique identifier of each request message. The response message MUST contain the same Request-ID as that of the corresnponding request message to make correspondence between requests and responses.

In order to distinguish a PePP message from the subsequent ones, a PePP message MAY contain a Content-Length header field. If a PePP message has a Content-Length header, its value MUST match the exact data size of the message body.

In the client connections, all the PePP requests defined in this document can be issued.

(c) Closing a Connection

In order to close a PePP connection, a LOGOUT request MAY be sent by either ends of the connection. When a client or server receives a LOGOUT request, it MUST send 200 OK response if it is not to send another request to the connection peer. A client or server which has issued a LOGOUT request SHOULD wait an OK response before closing the connection.

A client or server MAY close the connection without issuing a LOGOUT request in the case it encounters an abnormal status. For instance,

the connection MAY be closed without any notice in the following cases.

1) The message border in the connection is broken because of a wrong Content-Length specified.
2) The size of the data currently receiving has exceeded the Max-Content-Length of the connection.
3) Time-out.

(d) Backup Connections

A client MAY request to establish more than one connections as "backups" if necessary. A backup connection is established by the same procedure as the main connection. It is typically considered necessary when the client is to send or receive data larger than the Max-Content-Length value of the main connection. The purpose of backup connections is to avoid latency caused by sending huge data through relatively slow connections.

A client MUST request to establish a backup connection when it is to send a larger message body than the Max-Content-Length values of existing connections. If the data size is less than one of the Max-Content-Length values of the existing connections, the client SHOULD NOT request a new connection. A LOGIN request for a backup connection MUST have a Backup-For header field specifying the connection ID of the main connection.

Although a server cannot request to open a new client connection, the server can issue a CALLBACK request through the main connection to ask the client to open a new connection. The CALLBACK request MAY contain information of the new address to be connected. If the server sends a message with a content larger than the Max-Content-Length values of existing connections, it MUST send a CALLBACK request to the main connection and wait for a new connection. The client MAY refuse the CALLBACK request.

Backup connections MAY be closed by either end of the connection if it is considered no more necessary. Backup connection SHOULD be closed by LOGOUT requests.

3.3.2. Server Connections

A PePP connection in the "server" mode, or a PePP server connection, is a TCP connection between servers. We use a term "Target Server" in contrast with "Home Server" to designate the remote server which the client cannot connect directly.

If the Home Server of a client receives requests destined for another Target Server, it MUST establish a PePP server connection with the Target Server and forward the request to it. Unlike PePP client connections, only the initiator of the connection can issue requests in general and it MAY close the connection if it is judged not to be necessary any more.

(a) Establishing a Connection

When a server is to establish a PePP server connection, it MUST try to look up a DNS SRV record for the "impp" service on the "tcp" protocol prior to looking for an A record to locate another server.

After locating the other server, the initiating server opens a TCP connection to the other and sends a LOGIN request to start authentication process. In the LOGIN request, the initiating server specifies information about the authentication process, a connection mode, and the maximum content size it wishes to receive. For the server connection, the value "server" MUST be specified as a value of Connection-Mode header field.

The allowed SASL Mechanisms for PePP server connections are CRAM-MD5 [CRAM-MD5] and EXTERNAL [SASL]. PLAIN is disallowed because it seems unnecessary if the servers could authenticate each other by the TLS mutual authentication, and this is very likely.

1) EXTERNAL - It uses TLS client authentication, and can be specified only if TLS client authentication is supported.
2) CRAM-MD5 - It MAY be accepted depending on the server policy.

CRAM-MD5 is only for an experimental use because sharing the secret passwords between different domains seems to be undesirable. It is STRONGLY RECOMMENDED to upgrading to a secure transport by issuing a STARTTLS request prior to the LOGIN request.

(b) PePP Messages

A PePP server connection is mainly used to exchange request and response messages between different domains. Unlike PePP client connections, server connections are one-way connections, i.e. only the initiating server of the connection can issue request messages except for a LOGOUT request.

Like client connections, every PePP messages MUST contain a Request-ID which is a unique identifier of each request message, and every PePP messages MUST have a Content-Length header field whose value is the exact data size of its message-body.

In the server connection, request messages for administration use is not allowed. Only the following request messages can be issued: LOGIN, LOGOUT, SUBSCRIBE, UNSUBSCRIBE, REQUESTNOTIFY, NOTIFY, PULL, SEND, PING, STARTTLS, CONNECT.

(c) Closing a Connection

Like PePP client connections, the either side of the connection MAY issue a LOGOUT request MAY to close it. When one of the connection peers receives a LOGOUT request, it MUST send '200 OK' response if it is not to send another request to the connection. The connection peer which has issued a LOGOUT request SHOULD wait an OK response before closing the connection.

Same as the case of client connections, each connection peer MAY close the connection without issuing a LOGOUT request in some abnormal status.

(d) Backup Connections

A server MAY request to establish more than one connection to the same server at any time if necessary. So, a server MAY have one or more connections with the same Max-Content-Length value.

As we may assume a PePP server can accept a LOGIN request from other servers, we do not distinguish "backup" connection from the "main" one. They are independent connections.

3.3.3. Direct Connections

PePP has a mechanism to establish a virtually direct connection between clients. This is an OPTIONAL feature in PePP.

A "Direct Connection" is a PePP Connection in the "direct" mode. The Direct Connection is the virtual connection between the clients and implemented using the Client Connection and the Server Connection.

The Direct Connection is designed for providing the end-to-end security to PePP, especially a private conversation channel for IMs between clients in order not to be tampered by even the administrators of the servers.

(a) Establishing a Connection

When the client is going to establish to Direct Connection, first of all, the initiator client opens a new Client connection to its home server, and issues a LOGIN request to identify itself. Then, the initiator issues a CONNECT request, which asks the Home Server to provide a "raw" TCP socket over the current connection.

The home server holds this TCP connection and opens another TCP connection to the Target Server. The way for discovering the Target Server is same as Server Connection described above.

After LOGIN, the Home Server issues CONNECT request to ask "raw" TCP connection to the Target Server. Then, the Target Server issues a CALLBACK request to one of the 'receiver' clients which has asked IM delivery by the RECEIVE request (See Section 3.5).

Having established a brand-new TCP connection with the client as a result of the CALLBACK request, the Target Server responds to the Home Server '200 OK' response, and the Home Server also responds '200 OK' response to the initiator client.

Finally, the initiator client issues STARTTLS command, which is directly sent to the target client, and they can send and receive messages securely over this virtual connection.

(b) PePP messages

The Direct Connection behaves like as a Client Connection does, but only SEND, PING and LOGOUT requests are allowed.

(c) Closing a Connection

Both of the clients can issue a LOGOUT request to terminate the Direct Connection. The client which has issued a LOGOUT request SHOULD wait to receive the response.

(d) Backup Connections

Because the separate Direct Connections MAY be established between the same two clients, backup connections for the direct connection will not be necessary.

3.4. Subscription Model

As stated in the previous sections, a PePP client MUST subscribe to

Presence Information in the Target Server via its Home Server. The Home Server and Target Server are generally distinct servers while they might happen to coincide (Fig. 3). If a client has a subscription to a resource at a Target Server, the subscription information is stored at the client, its Home Server, and the Target Server.

This subsection describes the behavior of these components in relation with Presence Information subscription.

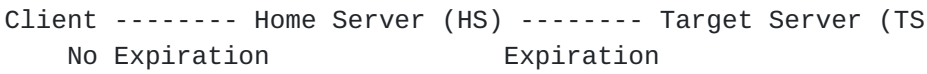

Fig.3: Subscription Model

3.4.1. Behavior of Clients

The PePP client connection is expected to be persistent until the client sends LOGOUT to the HS or some failure occurs. If the connection is in an abnormal status, for instance the response of PING request cannot be received, the client SHOULD disconnect the connection and try to reconnect in a certain amount of time and reissue all the SUBSCRIBE requests.

3.4.2. Behavior of Home Servers

Because a PePP server connection is not necessarily persistent, the servers on the connection peers cannot detect another one's failure by watching the connection. As a solution to this problem, PePP utilizes the expiration model for subscription. Thus, a subscription will expire unless it is renewed by another subscription request within a certain amount of the associated duration time.

A Home Server MUST store and manage all the subscription information of its clients possibly to the other servers. More precisely, a HS MUST watch all the requests from the clients to subscribe or unsubscribe resources and all the cancel notification from the TSs, and store the up-to-date information about each subscription, which consists of the target resource, subscription ID, duration, and the client connection ID. Moreover, in order to save the traffic on the client connections, the HS MUST renew all its clients' subscriptions regularly on behalf of the clients.

If the HS detects a client connection has been lost, it MUST send requests to Target Servers to unsubscribe all the current subscriptions from the relevant client.

The duration for subscription should be considerably long in order to reduce the traffic of renew messages.

3.4.3. Behavior of Target Servers

The Target Servers MUST keep and manage the current subscription information in order to issue change notifications about the target resources. Because subscriptions are subject to expiration as stated above, a server SHOULD remove the subscription information unless it is renewed within a certain amount of duration time.

When the TS detects an error of the notification requests it has issued, the subscription information which caused the failed notification MUST be removed. When the server removes some subscription information, it MUST send a notification to the relevant subscriber asking her/him to retry subscription.

3.5. Instant Messaging Service

3.5.1. Basic Architecture

As PePP was originally developed for the Presence Service which requires minute control of presence publication, its basic Instant Messaging (IM) feature is designed similar to PePP's subscription and notification mechanism.

A user's INSTANT INBOX in PePP is a PePP resource to be addressed when an IM is sent to the user. The receiver's client issues a RECEIVE request to the INBOX resource to register itself as a destination to forward the IMs it receives. Then the client connection is registered as a 'receiver' connection. An IM is delivered by SEND requests. When the INSTANT INBOX receives an IM, the server issues a new SEND request to forward the IM to the 'receiver' connection (see Fig.4).

In PePP, the INBOX address of a receiver is not uniquely determined by her/his PePP address. Actually, the INBOX address may be same as the receiver's PePP address and may be different. Thus, the INBOX address MUST be contained in the user's presence information. The PePP client SHOULD send IMs for this address in order to be received by the intended recipient unless other address has been specified in an out-of-band manner.

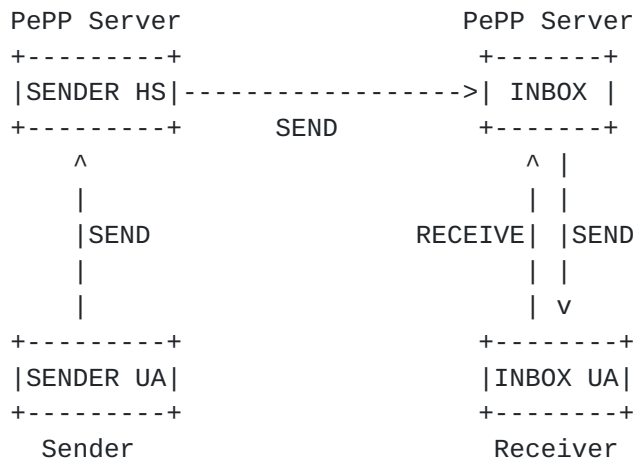

Fig.4: Basic Architecture of PePP IM Service

3.5.2. IM Conversation

Although an IM can be used as a single one-way message, the typical usage in the IM Service is a conversational one, i.e. two or more users exchange IMs in a 'chat' style. When a user wants to talk with a buddy, she directs an IM application to open a window for conversation, and uses the window for a talk with the buddy. She may open several conversation windows to talk with some of her buddies at the same time.

To realize this, each SEND message MUST have a Conversation-ID header field to identify the conversation it belongs. The Conversation-ID field MUST have a globally unique value like a Message-ID, which is also a globally unique identifier of the SEND message given by the client. The initial client to start the conversation thread with others MUST assign a value of Conversation-ID. It MAY reuse its Message-ID as the Conversation-ID.

3.5.3. Multiparty Conversation

While PePP's basic IM mechanism is for one-to-one conversation, it can be extended to a simple multiparty IM conversation. If a participant of an already established IM conversation wants somebody to join, he send a SEND request message to the user with the current Conversation ID and the Reply-To header whose value is a list of recipients. Then the invited user joins the conversation by sending his message with the specified Conversation ID to all the recipients described in the Reply-To field.

Obviously, this method of multiparty conversation has two problems.

One is a scalability problem if the number of participants increases, and the other is it does not provide a means to delete the recipient when one participant quits the conversation. But, we think usual IM conversation is shared by very small number of users, and it will be closed in a short time. It is expected that it would not cause so much practical problems.

3.6. Character and Content Encoding

For character encodings, PePP clients MUST accept the UTF-8 encoding of the ISO/IEC 10646 (UCS-4) character set, and MUST NOT cause errors by handling them. The user agent MUST display the content of presence information, instant messages, and other messages for at least US-ASCII part of UTF-8 encoding.

Content or character encoding method is declared in 'charset' attribute in Content-Type header as in the example below.

```
Content-Type: text/plain; charset=UTF-8
```

For the content type which has 'charset' as its attribute, specifying encoding method in 'charset' is STRONGLY RECOMMENDED. If the content type is text/xml, character encoding MAY be specified in the XML declaration as in the following example. In this case, the XML declaration is used.

```
<?xml encoding='ISO-2022-JP' ?>
```

PePP servers and proxies MUST NOT cause an error for arbitrary content of presence information and instant messages in any content encodings. PePP servers and proxies MUST deliver the content of presence information and instant messages to the targets.

4. Considerations Regarding IMPP Requirements

The IMPP Requirements document [Reqts] describes that an interoperable and widely-deployable standard protocol for IM/Presence must be scalable, secure, and appropriate for use with wireless devices. This section contains the considerations about PePP's strength and weakness for these criteria. Additionally, we give a note on the requirement for the separability of IM and Presence services.

4.1. Scalability

There are many aspects to scalability issues. We have considered the following features in the design of PePP;

(1) the option to distribute server load over multiple servers,
(2) the avoidance of processing bottlenecks where a delay in processing one message blocks other messages.
(3) the avoidance of connection bottlenecks where a single huge message blocks many smaller messages.

As a means to reduce the load of the servers, PePP allows any command to be redirected to an alternate server. This enables various strategies for dynamically allocating server resources and load balancing.

To avoid the processing bottlenecks, PePP allows the responses to be received in the different order from that of requests by utilizing the Request ID in the messages so that the response can be matched with the corresponding request.

There are two typical solutions to avoid the connection bottlenecks, data chunking/interleaving on a single connection, or the creation of multiple parallel connections. PePP has adopted the latter because it seems reasonable to open a new connection to send a huge data during an IM conversation. Another reason is that command-level chunking/interleaving is difficult in PePP as its command syntax is based on HTTP.

4.2. Security

We have adopted the following security model for the IM/Presence services in PePP.

4.2.1. Trust model

We assume the network is not trustworthy at all.

Once authenticated each other, client-server and server-server relations are considered to be fully trustworthy. That is, the servers are trustworthy in the sense that;

* the servers disclose presence information and/or IMs to authorized parties only.
* the servers distribute presence information and/or IMs uncorrupted.
* the IM servers relay IMs only from authorized senders.

However, even though a user could trust the home server and servers

are mutually trustworthy, other servers may be less trustworthy for the user. For instance, the other domain might not support any transport security for the Client connections. The current specification of PePP does not have a mechanism of knowing the security level of other domains.

4.2.2. Transport Security

As we cannot assume the network is trustworthy, IM/Presence services require transport security to prevent tapping and tampering of messages. PePP adopts TLS for this purpose. The Server Connections are STRONGLY RECOMMENDED to be encrypted using TLS.

4.2.3. Confidentiality of Presence Information

Presence information may be shared with an indefinitely large set of WATCHERS. Thus, end-to-end content encryption for each subscriber is too costly and impractical. If transport security is assumed, it is reasonable to provide no further content encryption. We believe access control preferences for presence information to provide an acceptable level of privacy control in PePP.

4.2.4. Integrity of Presence Information

Under the trust model stated above, we trust all servers to not corrupt or alter presence information. Thus, PePP does not provide any additional mechanism to protect the integrity of presence information beyond the TLS transport security.

Obviously, this is a weakness of PePP. We cannot say there is no more threat of the "Man-in-the-Middle" atack. To avoid this, we have to provide a means to put a digital signature on presence information. Because presence information in PePP is a set of individual sections, a digital signature is needed to each sections individually but this might be costly. So, it might be desirable to merge some sections together and sign on the merged sections.

Even if digital signature is available, there is another threat of the replay attack. But, if a timestamp could be included at the time of digital signature, it would be helpful to avoid the apparent attacks while this is not a perfect solution.

4.2.5. Confidentiality of IM

As PePP provides a basic functionality for transport security using TLS, IMs can be securely transported on the wire. However, only this does not provide the end-to-end security. For instance, a malicious server administrator can read the content of IM conversation.

To provide the end-to-end security, PePP has an optional feature to establish a virtually direct connection between the clients which can be encrypted by TLS entirely. This assumes that the both clients have their digital certificates for their identities. Once such a direct connection is established, the clients can talk completely securely without so much overhead.

We once considered the adoption of the message-level encryption standards such as S/MIME or OpenPGP. However, these standards impose the clients too much overhead, and seems to be impractical especially for the mobile devices. Moreover, if we assume transport security, this also implies the inefficiency of double-encryption.

4.2.6. Access Control

PePP's section mechanism provides fine-grained access control over published presence information. A publisher can specify exactly which sections any particular party will see. For example, some WATCHERs can be shown IM presence but not shown cell phone presence.

Moreover, by using the same section name for different sections, WATCHERs can be show different values for the same fields. We call this feature "personae", and think it is very useful for privacy-sensitive applications.

4.3. Wireless

Wireless devices usually have limited computing and communication resources. As a result, more concise protocols are better and binary formats can be most efficient. However, PePP has adopted text-based formats for readability, extensibility, and ease of debugging.

PePP's section-based presence format offers opportunities to reduce the size of tranferred content. For example, PePP allows selective subscription, which allows SUBSCRIBERs to receive only an interesting subset of all presence information sections. We think this selectivity is especially desirable for a wireless environment.

4.4. Separability of Services

PePP is designed as an integrated protocol for both IM and Presence Services. However, the requirements document seems to require that the protocol MUST allow separation of these services;

> 2.1.1. The protocols MUST allow a PRESENCE SERVICE to be available independent of whether an INSTANT MESSAGE SERVICE is available, and vice-versa.

Although PePP is a single protocol for IM and Presence Services, we think it allows the two services to be provided separately. As PePP is originally designed for Presence Service, PePP is applicable as a protocol for the Presence Service without IM. A presence section is generic for various communication means other than IMs, and a section for IM contains a URI as an IM address. This feature allows IM Service to be a distinct one from the Presence Service.

If we could disable the functions in PePP for Presence Service, it seems to be able to provide IM Service only. More concretely, functionality for establishing the PePP connections (LOGIN, STARTTLS, CALLBACK, CONNECT, and LOGOUT) and that for IM exchange (SEND and RECEIVE) seem to be sufficient to provide the IM Service in PePP. We assume that the IM addresses are available in an out-of-band manner.

5. PePP Messages

5.1. Message Overview

The message format for the PePP protocol basically follows the formats for HTTP/1.1 [HTTP1.1]. Thus, a message consists of a start line, zero or more header fields, and possibly a message-body. A start line is either a request line or response line. A request line contains a PePP command, a PePP Resource as a target resource, a Request ID of the request itself, and a PePP Version identifier.

PePP command details are described in Section 7.

Headers are defined in Section 6. Headers defined here MAY appear at most once in a PePP message unless otherwise stated. Headers in PePP messages which are not defined in this specification MUST be ignored except for the case of SEND messages.

A message-body conveys a content of presence information and instant messages. We use XML as a syntactic framework for the data format of presence information. MIME format is used when multiple presence sections are packed into a single message. MIME is also used for IMs.

The PePP Version identifier used for PePP messages in this spec is "PePP/0.5" at present.

5.2. PePP Addresses

A PePP Resource is represented as a form similar to the HTTP URL defined in HTTP/1.1 [HTTP1.1]. That URI is called the PePP Address of the resource. The same namespace is used for both Presence Service and IM Service in PePP. Because PePP is designed for PePP services, we use "pepp" for the protocol scheme name in the URL namespace. The syntax of PePP Addresses is defined as follows.

```
PePP-Address      = "pepp:" "//" host [":"port]] abs_path

host              = <FQDN or IP address (in dotted-decimal form),
                    as defined by Section 2.1 of RFC 1123>

port              = 1*DIGIT

abs_path          = <defined in [URI]>
```

In the PePP addresses, the local namespace can be extended utilizing paths like HTTP URL by the domain administrator or the owner of the resources for Presence Information or the INBOX address for IMs.

PePP utilizes the PePP Address of the user's top resource as an identifier of the user, which is used in the From header of a PePP request.

5.3. Message Syntax

The format of PePP messages is defined as follows, which is described in an augmented Backus-Naur Form (BNF) used in [HTTP1.1].

```
PePP-Message      = PePP-Request | PePP-Response
PePP-Request      = PePP-Command SP Request-ID SP PePP-Version CRLF
                    *(( PePP-Header ) CRLF)
                    CRLF
                    [message-body]

PePP-Response     = PePP-Status-Line
                    *(( PePP-Header ) CRLF)
                    CRLF
                    [message-body]
```

```
PePP-Status-Line  = PePP-Version SP Request-ID SP Status-Code SP
                    Reason-Phrase CRLF

PePP-Version      = "PePP" "/" 1*DIGIT "." 1*DIGIT

Request-ID        = token


PePP-Command  = LOGIN           ; section 7.1
              | LOGOUT          ; section 7.2
              | SUBSCRIBE       ; section 7.3
              | UNSUBSCRIBE     ; section 7.4
              | REQUESTNOTIFY   ; section 7.5
              | CHANGE          ; section 7.6
              | CANCEL          ; section 7.7
              | FETCH           ; section 7.8
              | NOTIFY          ; section 7.9
              | PULL            ; section 7.10
              | SEND            ; section 7.11
              | RECEIVE         ; section 7.12
              | CALLBACK        ; section 7.13
              | REDIRECT        ; section 7.14
              | SETACL          ; section 7.15
              | GETACL          ; section 7.16
              | CREATESECTION   ; section 7.17
              | DELETESECTION   ; section 7.18
              | PING            ; section 7.19
              | STARTTLS        ; section 7.20
              | CONNECT         ; section 7.21


PePP-Header   = From                ; section 6.1
              | Connection-Mode     ; section 6.2
              | Max-Content-Length  ; section 6.3
              | Subscription-Mode   ; section 6.4
              | Subscription-ID     ; section 6.5
              | Regarding           ; section 6.6
              | Change-Mode         ; section 6.7
              | Cancel-Type         ; section 6.8
              | Duration            ; section 6.9
              | Last-Modified       ; section 6.10
              | Section-ID          ; section 6.11
              | Section-Name        ; section 6.12
              | Location            ; section 6.13
              | Content-Type        ; section 6.14
              | Content-Length      ; section 6.15
              | Auth-State          ; section 6.16
              | SASL-Mechanism      ; section 6.17
```

```
                  | Message-ID          ; section 6.18
                  | Conversation-ID     ; section 6.19
```

Status-Code and Reason-Phrase are described in Section 8.

6. PePP Headers

6.1. From

The From header field contains the PePP address of the requesting entity. The From header MUST be included in all requests transported through the PePP server connections. If a server receives a request without the From header through one of a server connection, the server MUST return 400 Bad Request error. Requests only used in PePP client connections MAY not have this header.

```
    From = "From" ":" PePP-Address
```

6.2. Connection-Mode

The Connection-Mode header field is included in LOGIN requests to indicate the mode of the connection. The value can take one of the two string tokens.

```
    Connection-Mode = "Connection-Mode" ":" ("server" | "client" |
                                             "direct")
```

- server
  This value indicates the connection is requested in the "server" mode.

- client
  This value indicates the connection is requested in the "client" mode.

6.3. Max-Content-Length

The Max-Content-Length header field is included in LOGIN requests/responses and CALLBACK requests and indicates the size of an acceptable message body by the connection in decimal number of octets. The syntax is defined the same as in HTTP/1.1 [HTTP1.1].

```
     Max-Content-Length = "Max-Content-Length" ":" 1*DIGIT
```

6.4. Subscription-Mode

The Subscription-Mode header field which appears in the SUBSCRIBE request specifies the mode of the requesting subscription. The value can take one of the three; notify, pull, renew.

```
Subscription-Mode = "Subscription-Mode" ":" ("notify" | "pull" |
                                              "renew")
```

- notify
  If the client requests to be notified when a change occurs in the target resource, this mode is specified. This is default.

- pull
  The client tells the server that it would not like any changes to be notified. When the client wants to get the content of the resource in the Pull mode, it sends a PULL request to the server explicitly.

- renew
  This mode is used in order to renew the subscription specified by the Subscription-ID. The response caused by the subscription request with this mode MUST NOT have a message body.

6.5. Subscription-ID

The Subscription-ID header is used to specify the identifier of the subscription of concern in the request or the response. The server MUST specify this header field and value in the response to the SUBSCRIBE request. The client uses this value in the subsequent request to specify the subscription.

```
Subscription-ID = "Subscription-ID" ":" token
```

The value of Subscription-ID MUST be uniquely assigned at least modulo PePP resource.

6.6. Regarding

The Regarding header is used in the SUBSCRIBE, FETCH or NOTIFY requests. The value can take one of two string tokens.

```
Regarding = "Regarding" ":" ("value" | "control" )
```

If the Regarding header field appears in SUBSCRIBE or NOTIFY requests, it designates the kind of the subscription or notification. The "value" and "control" in this field specifies the subscription or

notification is regarding the Presence Information and Subscribers Information respectively. If the Regarding header appears in a FETCH request, it means the kind of information the request tries to fetch. The meaning of the two values are same as in SUBSCRIBE and NOTIFY requests. The default value is "value".

6.7. Change-Mode

The Change-Mode header is used in the CHANGE request to specify the behavior of the request. The value can take one of four string tokens.

```
Change-Mode = "Change-Mode" ":" ("lease" | "permanent" | "renew" |
                                 "revert" )
```

- o lease
  This mode is used to set or change the lease value of the presence section. It resets the lease timer of the section, and causes to send change notifications to the subscribers.

- o permenent
  This mode is used to set or change the permanent value of the presence section. If there is no lease value, it causes to send change notifications to the subscribers.

- o renew
  This mode is used to renew the lease value of the presence section. It resets the lease timer of the section, but does not cause any notifications.

- o revert
  This mode is used to remove the lease value of the presence section and revert to its permanent value. It causes to send change notifications to the subscribers.

6.8. Cancel-Type

The Cancel-Type header is used in CANCEL requests and the NOTIFY requests caused by the CANCEL requests.

```
Cancel-Type = "Cancel-Type" ":" ("cancel" | "retry")
```

When a subscription is CANCELed, a NOTIFY request is issued to the WATCHERS in order to specify the expected action of the receiver client. If the server wants to direct the WATCHER client to retry subscription, the "retry" value MUST be set in the Cancel-Type header

field. If the server wants to state the client not to retry to subscribe, the "cancel" value MUST be set in this field. The client SHOULD NOT subscribe to the same resource if the subscription was canceled with the value "cancel" in the Cancel-Type header.

The Cancel-Type header is used in the CANCEL requests to specify the Cancel-Type header in the NOTIFY request caused by it. If this header is omitted in the CANCEL request, the value of "cancel" is used.

6.9. Duration

The Duration header field specifies a lifetime of the lease value of the presence section in an integer second count if it is used in a CHANGE request or its response. The response for the CHANGE request with the Change-Mode 'lease' and 'renew' MUST contain the Duration header field. Such a CHANGE request MAY contain the Duration header as the client's request, but the server MAY ignore the value based on its policy. The client MUST use the specified value in the response as the duration for the presence section.

The Duration header is also used in SUBSCRIBE request issued by the Home Server to specify a lifetime of the subscription. The response for the SUBSCRIBE request MUST contain the Duration header that specifies the duration of the subscription. The Home Server MUST use the specified duration value in the response even if it specified a different value in the SUBSCRIBE request.

```
Duration = "Duration" ":" 1*DIGIT
```

6.10. Last-Modified

The Last-Modified header field specifies the date/time of the latest change of the transported content. It is specified by the server.

For Presence Information, each presence section has a last-modified value, and the value is changed in the following four cases; a) a CHANGE request changes the lease value, b) the lease value expires and the value changes to the permanent value, c) the permanent value is changed when the lease value is not set, d) the lease value is removed. The generated NOTIFY request message MUST have the Last-Modified header field containing this value.

The response of a SUBSCRIBE or FETCH request MAY have MIME Multipart content with the multiple presence sections. In this case, the

Last-Modified header MUST appear as one of the MIME-part-headers of each body part of the multipart entity.

The date/time format is specified as follows. It is one of the format specified in ISO 8601 [ISO8601].

```
Last-Modified = "Last-Modified" ":" date "T" time "Z"

date = 4DIGIT "-" 2DIGIT "-" 2DIGIT
     ; year-month-day
time = 2DIGIT ":" 2DIGIT ":" 2DIGIT
     ; hour:minute:second (00:00:00 - 23:59:59)

Example:   1999-12-08T18:05:23Z
```

6.11. Section-ID

The Section-ID header field specifies the unique identifier of the presence section. When a presence section is to be created, the CREATESECTION request is issued by the client and the request MUST include this header. Its value is created by the client, and the uniqueness of the value is checked by the server. The CHANGE and DELETESECTION requests MUST contain this header as well. This header MAY also appear in the FETCH and CANCEL requests and responses to the FETCH requests. Section IDs are not to be shown to WATCHERS.

```
Section-ID = "Section-ID" ":" token
```

6.12. Section-Name

The Section-Name header field specifies the section name of the presence section. Section names are used by WATCHERS to specify the presence sections. When a presence section is to be created, the CREATESECTION request MUST include this header and the value.

```
Section-Name = "Section-Name" ":" token
```

6.13. Location

The Location header field specifies the PePP resource to be redirected. The REDIRECT request and the NOTIFY request caused by it MUST include the Location header.

```
Location = "Location" ":" PePP-Address
```

6.14. Content-Type

The Content-Type header field indicates the media type of the message body sent to the recipient. The syntax of the media type is defined the same as in HTTP/1.1[HTTP1.1].

As stated in section 3.6., if the media type has 'charset' attribute, specifying encoding method in 'charset' attribute is STRONGLY RECOMMENDED.

```
Content-Type = "Content-Type" ":" type "/" subtype *(";" parameter)
```

6.15. Content-Length

The Content-Length header field indicates the size of the message body, in decimal number of octets. The syntax is defined the same as in HTTP/1.1 [HTTP1.1].

```
Content-Length = "Content-Length" ":" 1*DIGIT
```

6.16. Auth-State

The Auth-State header specifies the status of authentication process in the LOGIN request.

```
Auth-State = "Auth-State" ":" ("init" | "continue" | "abort")
```

6.17. SASL-Mechanism

The SASL-Mechanism header specifies the SASL mechanism in the LOGIN request or the response to the LOGIN request. When used in the request, one SASL mechanism the client wants to use MUST be specified. When used in the response, one or more mechanisms which the server supports MAY be specified.

```
SASL-Mechanism = "SASL-Mechanism" ":" mechanism *(LWS mechanism)
```

6.18. Message-ID

The Message-ID header specifies the identifier of each IM, which distinguishes the message from others. The client MUST generate the Message ID unique to the PePP address for each IM. This header MUST appear in a SEND request.

Message-ID = "Message-ID" ":" token

6.19. Conversation-ID

The Conversation-ID header is used in the SEND request to identify the conversation channel shared by the participants of IM exchange. Here, a 'conversation channel' means a virtual channel which consists of a thread of the IM conversation. When a client replies to an IM in its same conversation channel, the SEND request for the reply MUST have the Conversation-ID header with the same value.

Conversation-ID = "Conversation-ID" ":" token

7. PePP Methods

This section describes the methods used in the PePP messages. We use the following notation to specify the allowable modes of connections and the directions of requests for each method.

s->s : Server Connections.
c->s : Client Connections, Client-to-Server direction.
s->c : Client Connections, Server-to-Client direction.
c->c : Direct Connections.

7.1. LOGIN

7.1.1. Command

"LOGIN"

7.1.2. Direction

s->s
c->s
c->c

7.1.3. Headers

From
Auth-State
SASL-Mechanism
Connection-Mode
Max-Content-Length

7.1.4. Description

In order to establish the PePP connection, the initiator client or server MUST issue a LOGIN request to the other peer server to start authentication process. The Connection-Mode header indicates the mode of the required connection. When a client logins its Home Server, it MUST LOGIN the server in the "client" mode. When a server tries to open a connection with servers in the different domains, it MUST LOGIN the target server in the "server" mode.

If the authentication process is successfully fulfilled, a PePP connection is established between the initiator and the target. Otherwise, the initiator MUST not send any commands. If it try to send other command in that case, the target server MUST return 401 Unauthorized error.

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 "init" Auth-State value indicates the initial request in the process, the "continue" value indicates it is subsequent requests. The SASL-Mechanism header is used to exchange the SASL mechanism supported by the initiator and the target server.

Once a PePP connection is established, the initiator MUST NOT send another LOGIN request to the same connection. The initiator MUST NOT issue another LOGIN request with "init" Auth-State value in the midst of the authentication process. In either case, 406 Authentication Failed response is returned by the server.

The LOGIN request MAY be preceded by the STARTTLS request when the implementations support TLS for a secure PePP connection.

7.2. LOGOUT

7.2.1. Command

"LOGOUT"

7.2.2. Direction

s->s
c->s
s->c
c->c

7.2.3. Headers

7.2.4. Description

In order to terminate the currently established PePP connection, the either side of the connection SHOULD issue a LOGOUT request in a normal situation. The issuer of the LOGOUT request SHOULD wait its response to confirm the other peer is to send nothing.

7.3. SUBSCRIBE

7.3.1. Command

"SUBSCRIBE" SP PePP-Address

7.3.2. Direction

s->s
c->s

7.3.3. Headers

From
[Subscription-Mode]
[Subscription-ID]
[Regarding]
[Duration]

7.3.4. Discription

The SUBSCRIBE method is used to declare a subscriber's interest at a PePP resource. The success of SUBSCRIBE request to a resource causes a change in the Subscribers Information at the resource.

A SUBSCRIBE request MAY include 'Subscription-Mode' header, whose possible value is "notify", "pull", and "renew". If the "notify" value is specified, any changes in the subscribed sections will cause a notification message from the server. If the "pull" value is specified, the server MUST NOT send notification messages for this subscription unless the subsequent REQUESTNOTIFY message requests otherwise. The "renew" value is only specified by the Home Server of the subscribers in the case of renewing the subscription specified by the 'Subscription-ID' header field.

There are two kinds of subscriptions regarding the information to be

subscribed; value and control. A SUBSCRIBE request MAY have 'Regarding' header to designate the kind of subscription. Possible values for this header are 'value' and 'control' respectively. The default is 'value', which means a usual subscription to the presence information.

If 'Regarding: control' is specified, the client subscribes to the Subscribers Information at the resource (see Section 11).

For a SUBSCRIBE request with 'Regarding: value' header field, the target server determines the permitted presence sections to be shown based on the ACCESS RULES of the target resource. The response for the SUBSCRIBE request MUST contain the presence information of the allowed presence sections unless it is a "renew" request. Even if the ACCESS RULE changes after the subscription, the currently shown set of presence sections will not change until the client issues another SUBSCRIBE request.

The response to a successful SUBSCRIBE request other than "renew" MUST contain 'Subscription-ID' header specifying the unique identifier of the subscription, and the 'Duration' header field specifying the amount of the duration time in seconds. The Home Server MUST maintain the subscription ID and the duration value in relation with the subscriber's connection ID, and MUST renew the subscription on behalf of the subscriber's client. The target server MUST NOT discard a subscription information before it expires in a normal situation. The Home Server MUST relay the response containing the subscription ID, but the client does not have to refer or specify any duration value.

The maximum number of subscription at a particular resource can be set. In this case, if the server receives SUBSCRIBE requests exceeding the maximum, it MAY return a '505 Too Many Subscription' error.

PePP does not offer any particular method to get the list of presence sections. So, one who wants a list of presence sections should issue a SUBSCRIBE request. While PePP does not offer any method to specify whether or not the pull mode subscription is allowed, an implementation MAY provide a method to disallow it in an out-of-band fashion.

7.4. UNSUBSCRIBE

7.4.1. Command

"UNSUBSCRIBE" SP PePP-Address

7.4.2. Direction

s->s
c->s

7.4.3. Headers

From
Subscription-ID

7.4.4. Description

The UNSUBSCRIBE method is used to terminate a previously established subscription. It MUST include a 'Subscription-ID' identifying the subscription to be unsubscribed. The absence of this header causes an error response. This method can be used only by the relevant subscribers.

7.5. REQUESTNOTIFY

7.5.1. Command

"REQUESTNOTIFY" SP PePP-Address

7.5.2. Direction

s->s
c->s

7.5.3. Headers

From
Subscription-ID
*(Section-Name)

7.5.4. Description

The REQUESTNOTIFY method is used to require change notifications on the presence sections specified by the Section-Name header fields through the already established subscription. The subscription is identified by the specified Subscription-ID, and, if the subscription does not exist, it will cause a '404 Subscription Not Found' error.

More than one Section-Name header fields MAY be specified at once. The REQUESTNOTIFY request always overwrite the subscription mode of each presence section. I.e. the presence sections specified in the

Section-Name headers change to the Notify mode and other sections change to the Pull mode. If no Section-Name header is specified, all sections become to be subscribed in the Pull mode.

7.6. CHANGE

7.6.1. Command

"CHANGE" SP PePP-Address

7.6.2. Direction

c->s

7.6.3. Headers

From
Section-ID
Change-Mode
Content-Length
[Duration]
[Content-Type]

7.6.4. Description

The CHANGE method is used to alter the content stored at a given resource. A CHANGE request MUST be targeted to a single presence section by specifying the Section-ID header. Section-ID header is mandatory and its absence causes a '400 Bad Request' error. A successful CHANGE request may cause notifications to subscribers who subscribe to the relevant presence section.

A CHANGE request MUST have a Change-Mode header field. The possible value is one of the four: "lease", "permanent", "renew", and "revert". A CHANGE request with the Change-Mode "lease" or "renew" MAY contain the Duration header field which specifies the client's request of the duration of the lease value.

If "lease" is specified, the content of the message body is set as the lease value of the presence section. The duration MAY be specified by the Duration header, but the client MUST use the value specified by Duration header of the response even if it is different. The successful CHANGE request resets the lease timer of the section and causes notification messages to subscribers. If the lease value is not renewed within the amount of the specified duration value, it expires and the section reverts to its permanent value.

If "permanent" is specified, the content of the message body is set as the permanent value of the presence section. If no lease value is set, it causes to send change notifications to the subscribers.

If "renew" is specified, the lease value of the presence section is renewed. It resets the lease timer with the duration specified by the Duration header field in the response. It does not cause any notifications.

If "revert" is specified, the lease value is removed and the value of the presence section reverts to its permanent value. It causes notification messages to subscribers.

7.7. CANCEL

7.7.1. Command

"CANCEL" SP PePP-Address

7.7.2. Direction

c->s

7.7.3. Headers

From
[Section-ID]
[Subscription-ID]
[Cancel-Type]

7.7.4. Description

The CANCEL method is used to direct the target resource to discard the subscription specified in the Subscription-ID header. This is only issued by the client of the PRESENTITY.

When a subscription is canceled by a successful CANCEL request, a NOTIFY request message reporting the cancellation is sent to the subscriber. If the CANCEL request contains the Cancel-Type header field (possible values are 'retry' and 'cancel'), the resulting NOTIFY request MUST contain the Cancel-Type header field with the same value. If the CANCEL request does not contain the Cancel-Type header, the resulting NOTIFY request MUST contain the Cancel-Type header with the value 'cancel'.

Even if the subscription to be canceled is in the Pull mode, such a

reporting NOTIFY message SHOULD be issued. However, in the case that the NOTIFY message is not delivered to the subscriber successfully, the subscriber may send a PULL request through the CANCELed subscription. In this case, the server MUST retrun the '404 Subscription Not Found' error.

If the CANCEL request specifies neither Subscription-ID nor Section-ID headers, all the subscription SHOULD be canceled at the target PePP resource. If the CANCEL request has the Section-ID header and does not include the Subscription-ID header, all the subscriptions in relation to the specified section SHOULD be canceled. If there the subscription specified by the Subscription-ID header does not exist, it MUST cause 404 Subscription Not Found error.

7.8. FETCH

7.8.1. Command

"FETCH" SP PePP-Address

7.8.2. Direction

c->s

7.8.3. Headers

From
Regarding
[Section-ID]

7.8.4. Description

The FETCH method is used to get presence information and/or control information in the targeted resource. This method is mainly used for control use by the owner of the resource.

FETCH requests can be targeted both to a resource and to a presence section contained in a resource. If the FETCH request contains the Section-ID header, the content of the specified presence section is returned. The response MUST also include the Section-ID header.

When targeted to an entire resource, and if the resource contains multiple sections, the contents of multiple sections are returned in a single response formatted in MIME multipart. Each part MUST contain the Section-ID header whose value is the Section ID of the corresponding section.

7.9. NOTIFY

7.9.1. Command

"NOTIFY" SP PePP-Address

7.9.2. Direction

s->s
c->s

7.9.3. Headers

From
Subscription-ID
Section-Name
Content-Length
Content-Type
[Regarding]
[Cancel-Type]

7.9.4. Description

The NOTIFY method is used (1) to report CHANGEs inside a subscription; and (2) to report CANCELs of subscriptions to the subscribers.

The NOTIFY request MUST include the Subscription-ID header to specify the subscription by which the notification is required. This specification does not specify the behavior of the receiver in the case the Subscription-ID is missing.

The target address of the NOTIFY request MUST be the address in the From header of the corresponding SUBSCRIBE request.

The Regarding header has the same value as specified in the corresponding SUBSCRIBE request. The default value is 'value'.

If a subscription at a resource is canceled by a successful CANCEL request, it causes the NOTIFY request to the subscriber. Such a NOTIFY MUST contain the Cancel-Type header field. If the corresponding CANCEL request contains the Cancel-Type header field with the value 'retry', the resulting NOTIFY request MUST contain the Cancel-Type header field with the same value. Otherwise, the NOTIFY request MUST contain the Cancel-Type header field with the value 'cancel'.

7.10. PULL

7.10.1. Command

"PULL" SP PePP-Address

7.10.2. Direction

s->s
c->s

7.10.3. Headers

From
Subscription-ID
[Section-Name]

7.10.4. Description

The PULL method is used to get the content of the resource which is currently subscribed by the same issuer of this message. The PULL request MUST contain 'Subscription-ID' header, and the value of this header MUST contain a valid subscription ID.

If the PULL request does not have a Section-Name header, the response contains all the disclosed sections encoded in MIME Multipart.

7.11. SEND

7.11.1. Command

"SEND" SP PePP-Address

7.11.2. Direction

s->s
c->s
s->c
c->c

7.11.3. Headers

From
Message-ID
Content-Length
Content-Type
[Conversation-ID]

[Reply-To]

7.11.4. Description

The SEND method is used to send arbitrary content to a targeted PePP address. This is usually used for IMs.

The SEND request MUST contain the Message-ID header whose value is the globally unique identifier of the message. The client MUST have responsibility for the uniqueness.

If the receivers are set by the RECEIVE requests at the target resource of the SEND request, the server MUST issue another SEND request with the same PePP Resource and header fields to the receiver connections. If the target resource has no receivers, the '502 Service Unavailable' error is returned.

When the client wishes to start a conversational IM exchange, the SEND request MUST contain the Conversation-ID header field whose value is a globally unique identifier to be shared by the participants of the conversation. Assume that a client has reveived an IM with the Conversation-ID header. If the client wishes to reply to it in the same conversation channel, it MUST specify the same Conversation-ID field in the reply SEND message.

The SEND request is REQUIRED to tranport any MIME entity. Thus, it MAY contain any legal MIME header which may not be defined here. The servers MUST forward the SEND 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 SEND message.

7.12. RECEIVE

7.12.1. Command

"RECEIVE" SP PePP-Address

7.12.2. Direction

c->s

7.12.3. Headers

7.12.4. Description

The RECEIVE method is used to specify the connection to forward the

delivered SEND message at the target resource. When the server receives the RECEIVE request, the Client connection of the issuer client is set as a 'Receiver' connection at the target resource.

More than one 'Receiver' connections MAY be set if other clients issue the RECEIVE request at the same resource. The server MUST manage the 'Receiver' connections of every resources in it, and, when it receives a SEND message targeted at the resource, it MUST issue the new SEND requests with the same PePP-Address and headers to its 'Receiver' connections.

7.13. CALLBACK

7.13.1. Command

"CALLBACK"

7.13.2. Direction

s->c

7.13.3. Headers

Max-Content-Length
[Location]
[Connection-Mode]

7.13.4. Description

The CALLBACK method is used by the server to ask the client to create a new "backup" Client Connection.

The server will use the newly established connection to send the message whose body size exceeds the Max-Content-Length values of the existing Client Connections.

The CALLBACK request MUST contain the Max-Content-Length header field to tell the required value for new connection.

The server MAY specify the Location header field to specify a different server location and/or port number to be called back from the client. If Location header value contains other than server and port number, rest part of PePP-Address will be ignored.

If the client accepts the request, it returns '200 OK' as the response and it MUST issue a LOGIN request to a newly opened TCP

connection to establish a "backup" Client Connection. If the client rejects the request, the client returns '402 Permission Denied' response.

The Target Server will use this request to ask the Target Client for creating "raw TCP connection", which provides the Direct Connection. In this case, the CALLBACK command MUST contain the Connection-Mode header field with the value "direct".

7.14. REDIRECT

7.14.1. Command

"REDIRECT" SP PePP-Address

7.14.2. Direction

c->s

7.14.3. Headers

Location

7.14.4. Description

The REDIRECT method is used to specify the address for redirection of the SUBSCRIBE or SEND request to the PePP resource. A successful REDIRECT request results in returning the 300 Moved Temporary response to the subsequent SUBSCRIBE or SEND requests. Established subscriptions at the time of the REDIRECT request are still alive as they were. PULL requests through the subscriptions MAY still be accepted. As the subscribers cannot know the target resource was REDIRECTed, the client MUST issue CANCEL request in order to tell the subscribers that the resource was REDIRECTed.

The destination resource of the redirection is specified in the Location header. The REDIRECT request without a Location header cancels the redirection settled before. Even if no redirection was settled, cancellation request is returned with 200 OK.

7.15. SETACL

7.15.1. Command

"SETACL" SP PePP-Address

7.15.2. Direction

c->s

7.15.3. Headers

Content-Type
Content-Length

7.15.4. Description

The SETACL method is used to specify the ACL at the targeted resource. The message body of the SETACL request is used as a new ACL. The format of the ACL is described in Section 12.

The owner of the resource, or a user specially authorized by the system administrator, can issue the SETACL requests.

7.16. GETACL

7.16.1. Command

"GETACL" SP PePP-Address

7.16.2. Direction

c->s

7.16.3. Headers

7.16.4. Description

The GETACL method is used to acquire the ACL at the targeted resource. The successful response contains the currently set ACL at the resource in its body part. The format of the ACL is described in Section 12.

The owner of the resource, or a user specially authorized by the system administrator, can issue the GETACL requests.

7.17. CREATESECTION

7.17.1. Command

"CREATESECTION" SP PePP-Address

7.17.2. Direction

c->s

7.17.3. Headers

Section-ID
Section-Name
Content-Type
Content-Length

7.17.4. Description

The CREATESECTION request is used to create a new presence section. Section-ID and Section-Name are mandatory headers and, if either of those is omitted, it causes 400 Bad Request error. The message body is set as a permanent value of this section.

The server checks the uniqueness of the Section-ID at the resource, and return 405 Section Already Exists if there already exists. This request does not contain any content of the presence section.

7.18. DELETESECTION

7.18.1. Command

"DELETESECTION" SP PePP-Address

7.18.2. Direction

c->s

7.18.3. Headers

Section-ID

7.18.4. Description

The DELETESECTION request is used to delete the specified presence section. The Section-ID header is mandatory. If absence, 400 Bad Request error is returned. If the section is still subscribed, a '407 Subscription Still Exists' error is returned.

7.19. PING

7.19.1. Command

"PING"

7.19.2. Direction

s->s
c->s
s->c
c->c

7.19.3. Headers

7.19.4. Description

The PING request is used by the server or the client to confirm that the PePP connection is alive. When this request is received, the receiver MUST return '200 OK' response.

7.20. STARTTLS

7.20.1. Command

"STARTTLS"

7.20.2. Direction

s->s
c->s

7.20.3. Headers

7.20.4. Description

A client or server MAY issue STARTTLS request to upgrade the existing TCP connection to the TLS [TLS] (formerly known as SSL) enabled one instead of using separate port for "secure" PePP connection. Implementations that support TLS SHOULD issue a STARTTLS request prior to issuing any other requests.

A TLS negotiation begins immediately after the "200 OK" response from the another peer. Once a STARTTLS command issued, the initiator MUST NOT issue further requests until a server response is received and the TLS negotiation is completed.

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 MUST NOT be used if the STARTTLS upgrading process fails to establish a fully strong encryption layer.

The implementation which does not support TLS SHOULD return the "501 Not Implemented" response. In this case, the client MUST authenticate itself in the following LOGIN process.

7.21. CONNECT

7.21.1. Command

"CONNECT" SP PePP-Address

7.21.2. Direction

s->s
c->s

7.21.3. Headers

From

7.21.4. Description

The CONNECT method is used to establish the Direct Connection between clients. The established connection will provide a private conversation channel for IMs.

The CONNECT request issued by a client intended to the other client is sent to the Home Server, and is forwarded to the Target Server by opening a new connection. Then, the Target Server issues a CALLBACK request to the destination client to tell the request from the initiator client (see Section 3.3.3 for details).

8. Status Codes

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

- 1xx: Informational - Request received, continuing process

- 2xx: Success - The action was successfully received, understood, and accepted

- 3xx: Redirection - Further action must be taken in order to complete the request

- 4xx: Client Error - The request contains bad syntax or cannot be fulfilled

- 5xx: Server Error - The server failed to fulfill an apparently valid request

8.1. 1xx

8.1.1. 100 Authentication Continued

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

8.2. 2xx

8.2.1. 200 OK

8.2.2. 201 Subscription Created

It appears in the response to a SUBSCRIBE request, reporting the successful result. In the subscription for 'Regarding: value' and 'control', the relevant content MUST be contained in the response.

8.2.3. 202 Section Created

It appears in the response to a CREATESECTION request, reporting the successful result.

8.3. 3xx

8.3.1. 300 Moved Temporary

The requested resource has been assigned a new URI temporarily and the requester SHOULD resend the request to the specified resource. The new URI MUST be given by the Location header field in the response. It depends on the server's policy to select the 300 or 301 response.

8.3.2. 301 Moved Permanently

The requested resource has been assigned a new permanent URI and any future references to this resource SHOULD use the returned URI. The new permanent URI MUST be given by the Location header field in the

response. It depends on the server's policy to select the 300 or 301 response.

8.4. 4xx

8.4.1. 400 Bad Request

The request could not be understood by the server due to malformed syntax. The client SHOULD NOT repeat the request without modifications.

8.4.2. 401 Unauthorized

The request requires user authentication. The client MUST authenticate itself by the LOGIN request.

8.4.3. 402 Forbidden

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

8.4.4. 403 Resource Not Found

The specified resource was not found at the server.

8.4.5. 404 Subscription Not Found

The Subscription specified in the Subscription-ID header was not found at the resource. This status code MAY appear in the response to the UNSUBSCRIBE, CANCEL, PULL requests and the SUBSCRIBE request in "renew" mode.

8.4.6. 405 Section Already Exists

The section specified in the Section-ID header of the CREATESECTION request already exists.

8.4.7. 406 Authentication Failed

The authentication process has been failed. The reason for it is one of the followings.

- The authentication process using the specified SASL-Mechanism was failed.
- The LOGIN request does not specify any SASL-Mechanism.
- The LOGIN request specifies inappropriate SASL-Mechanism.
- In the midst of the authentication process, the client tries to start another authentication process by specifying

'Auth-State: init'.

This response MAY contain a SASL-Mechanism header specifying the applicable SASL-Mechanism.

8.4.8. 407 Subscription Still Exists

The request has not been fulfilled because the subscription to the specified section still exists. This status code appears in the response to the DELETESECTION request. The client which has received this response MUST send a CANCEL request before requesting the DELETESECTION.

8.5. 5xx

8.5.1. 500 Internal Server Error

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

8.5.2. 501 Not Implemented

The server does not support the functionality required to fulfill the request.

8.5.3. 502 Service Unavailable

This status code is returned when the client issues the SEND request to the resource which any 'receiver' connection is not set.

8.5.4. 503 PePP Version Not Supported

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

8.5.5. 504 Gateway Timeout

The server forwarded the request to the specified server, but has not been received within the time that it was prepared to wait. the forwarded request has been timeout.

8.5.6. 505 Too Many Subscription

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

9. Presence Information Data Format

9.1. Overview

In PePP, Presence Information is encoded in Well-Formed XML without a DTD. Although any XML components MAY appear as a presence data, only the elements defined in this documents have a meaning.

Presence Information at a PePP resource is composed of a set of presence sections, each of which is contained in the <section> element. Each presence section has a unique identifier called Section ID, which is not to be shown to subscribers, and a section name which is sent to subscribers for the sake of selective subscription. However, both of section IDs and names are not included in the content of Presence Information. Instead, for each presence section, a display-name is contained in the content of PI to be displayed to subscribers. Display names are contained in the content of <display-name> element.

A typical example of Presence Information is availability of communication means, such as IM and telephone. Such information is contained in a <communication> sub-element. The <communication> element contains <address>, and <status> and <capability> sub-elements.

For IM, the <address> element contains the PePP address of the recipient. For the communication means where the target address can be expressed by a URL, the <address> element contains the URL (ex. <address>tel:+81-123-456-7890</address>). For other communication means, the <means> element contains supplementary information for the communication means.

Example:

```
<section>
  <communication>
    <address>pepp://pepp.org/Alice/iibox</address>
    <status><open><away/></open></status>
  </communication>
  <note>Out to Lunch.</note>
  <display-name>IM</display-name>
</section>
```

Example:

```
<section>
  <communication>
    <address>tel:+81-123-456-7890</address>
```

```
        <status><closed/></status>
      </communication>
      <display-name>Telephone at Home</display-name>
    </section>
```

The <communication> element MAY have a <capability> sub-element, which specifies the device capability of the communication means or the user's preference. Although this document does not specifies the concrete format of capability, we will allow to contain a URL where a capability for the device is stored in other future standard format as CONNEG or CC/PP [CC/PP].

9.2. Tag Descriptions

9.2.1. section

```
Tag:          section
Context:      top level
Appearance:   mandatory
Sub-elements: display-name note ( communication | principal )
Description:
  The section tag is top-level tag for presence sections.
```

9.2.2. display-name

```
Tag:          display-name
Appearance:   mandatory
Context:      section
Sub-elements: none
Description:
  Text to be displayed by the client UI.
```

9.2.3. note

```
Tag:          note
Appearance:   optional
Context:      section
Sub-elements: none
Description:
  Text to be handled as a short note in relation to the presence
  information.
```

9.2.4. communication

```
Tag:          communication
Appearance:   mandatory
Context:      section
```

Sub-elements: address communication-status capability
Description:
Information about communication means is contained. This element appears in a section element if the section is used to express status of a communication means. This element can have additional sub-elements.

9.2.5. communication-status

Tag: status
Appearance: mandatory
Context: section
Sub-elements: (open | closed)
Description:
Availability of the communication means.

9.2.6. open

Tag: open
Appearance: mandatory
Context: status
Sub-elements: (busy | away)
Description:
The open tag means that the communication device is available. It MAY contain other elements not defined here.

9.2.7. closed

Tag: closed
Appearance: mandatory
Context: status
Sub-elements: none
Description:
The closed tag means that the communication device is not available.

9.2.8. busy

Tag: busy
Appearance: optional
Context: open
Sub-elements: none
Description:
The communication device is available, but a communication request may not be noticed because the user is busy.

9.2.9. away

```
Tag:          away
Appearance:   optional
Context:      open
Sub-elements: none
Description:
  The communication device is available, but a communication
  request may not be noticed because the user is away from the
  device.
```

9.2.10. capability

```
Tag:          capability
Appearance:   optional
Context:      section
Sub-elements:  none
Description:
  If this element appears, the capability information of the
  corresponding communication device can be retrieved.
```

9.2.11. address

```
Tag:          address
Appearance:   mandatory
Context:      communication
Sub-elements: none
Description:
  The address of the communication device in the form of URI.
```

9.2.12. principal

```
Tag:          principal
Appearance:   mandatory
Context:      section
Sub-elements: principal-status
Description:
  Information in relation to the relevant principal is contained.
  The principal usually has various status information other than
  any communication means.  This tag is for such information.
```

9.2.13. principal-status

```
Tag:          status
Appearance:   mandatory
Context:      principal
Sub-elements: none
Description:
  Status information for the principal which may be used by
  the applications.
```

10. Subscribers Information

The owner or specially authorized user can get the information of the current subscribers at the resource. This is called Subscribers Information. The Subscribers Information is a list of subscription information at the resource, each of which contains the subscription ID, the subscriber's PePP address, the date of the subscription, and so on.

The Subscribers Information can be acquired by a SUBSCRIBE or FETCH request in 'Regarding: control'. Even if the Section-Name header appears in this request, it MUST be ignored. If the 'Subscription-Type: Notify' is specified, any change at the Subscribers Information will be notified.

The syntax of Subscribers Information is based on XML, and is defined by the following ABNF.

```
information       = "<information>" 1*subscription "</information>"
subscription      = "<subscription>" subscription-ID subscriber
                    created mode regarding "</subscription>"
subscription-ID   = "<subscription-ID>" token "</subscription-ID>"
subscriber        = "<subscriber>" PePP-Address "</subscriber>"
created           = "<created>" date "T" time "Z" "</created>"
mode              = "<mode>" ("notify" / "pull") "</mode>"
regarding         = "<regarding>" ("value" / "control") "</regarding>"
```

Here, the "created" element represents the date that the subscription was created. The format of the value is same as the Last-Modified header field specified in section 6.10.

The "mode" element represents the Subscription-Mode of the subscription which is defined in section 6.4 and 7.3.

The "regarding" element represents what property of the resource is subscribed. The value and its semantics is same as the Regarding header field in corresponding SUBSCRIBE request.

Example:

```
<information>
  <subscription>
     <subscription-ID>a1b2c3d4e5f6</subscription-ID>
     <subscriber>pepp://pepp.org/bob</subscriber>
     <created>2000-06-10T09:10:43Z</created>
     <mode>notify</mode>
     <regarding>value</regarding>
  </subscription>
```

```
    ....
  </information>
```

11. Access Control List

11.1. Overview

Access Control in PePP has two aspects; one is to decide allowing or rejecting the access request from the requesters, and the other is to select which presence sections should be disclosed to the SUBSCRIBERS. The user controlling a PePP resource can indicate how to handle the requests to it specifying Access Control List (ACL).

The PePP ACL specifies the action of the server at the time of receiving the following requests; SUBSCRIBE, SEND, FETCH, CHANGE, CANCEL, REDIRECT. For all of those requests but SUBSCRIBE, an allow-list and deny-list can be specified in the ACL. For SUBSCRIBE requests, a disclose-list can be specified for each section instead of an allow-list. This is a design decision based on our experience through the practice of presence services in our company.

By default, all requests are handled as they are denied, or nothing is disclosed, unless the system configuration allows. Because presence and instant messaging services are pretty sensitive to privacy issues, we took a safer side.

11.2. ACL Functions

11.2.1. SUBSCRIBE

For the SUBSCRIBE request, a basic action is 'deny' and 'disclose'. The ACL has a deny list and a disclose list for SUBSCRIBE.

- Deny-list is a list of principals whose requests are denied.
- Disclose-list is a set of lists, each of which is a list of principals allowed for each section.

A deny-list and a disclose-list appears once in the ACL, and the deny-list has a priority over the disclose list. In other words, a request from a principal matching the deny-list is rejected regardless of the disclose-list. The default action is also 'deny'.

Each section is specified by the Section ID and Section names does not appear in the ACL. This implies that it is possible to show more than one sections with the same name. The current specification does

not forbid this situation.

A 'default' tag MAY be assigned to one or more sections. The assigned section becomes a default section, which is shown when no section to be shown was specified explicitly.

11.2.2. SEND, FETCH, CHANGE, CANCEL, REDIRECT

For the SEND, FETCH, CHANGE, CANCEL, and REDIRECT requests, a basic action is 'deny' and 'allow', same as the conventional ACLs. The ACL has a deny list and an allow list for them.

- Deny-list is a list of principals whose requests are denied.
- Allow-list is a list of principals whose requests are allowed.

A deny-list and an allow-list appears once in the ACL, and the deny-list has a priority over the allow-list unless otherwise stated. In other words, a request from a principal matching the deny-list is rejected regardless of the allow-list. The default action is also 'deny'.

The priority can be reversed by specifying the 'priority-allow' tag. In this case, the allow-list has a priority over the deny-list and the default action becomes 'allow'.

11.3. Syntax of ACL

The syntax of ACL in PePP is based on XML as defined in this section. Command Names and Section IDs are case-sensitive.

The syntax is defined by the following ABNF.

```
acl             = "<acl>" subscribe *other-command "</acl>"
subscribe       = "<subscribe>" [deny-list] disclose-list "</subscribe>"
disclose-list   = "<disclose>" 1*section "</disclose>"
section         = "<section>" section-body "</section>"
section-body    = section-id (default / all / user-group-list)
user-group-list = *(user / group)
default         = "<default/>"
other-command   = "<command>" command-body  "</command>"
command-body    = command-name [reverse-priority] [allow-list] [deny-
list]
command-name    = "<name>" command-token "</name>"
command-token   = "SEND" / "FETCH" / "CHANGE" / "CANCEL" / "REDIRECT"
reverse-priority        = "<priority-allow/>"
allow-list      = "<allow>" (all / user-group-list) "</allow>"
```

```
deny-list       = "<deny>" user-group-list "</deny>"
user            = "<user>" user-id "</user>"
group           = "<group>" group-name "</group>"
all             = "<all/>"
```

Here, <group> tag is prepared for the future extension where a group of users can be specified as a principal.

Example:

```
  <acl>
    <subscribe>
      <deny>
        <user>pepp://pepp.fujitsu.com/suga</user>
      </deny>
      <disclose>
        <section>
          <secid>for-office1</secid>
          <user>pepp://pepp.fujitsu.com/sho</user>
          <group>group2</group>
        </section>
        <section>
          <secid>for-office2</secid>
          <group>group1</group>
          <group>group2</group>
        </section>
        <section>
          <secid>private</secid>
          <user>pepp://pepp.fujitsu.com/iwakawa</user>
          <user>pepp://pepp.fujitsu.com/sho</user>
        </section>
        <section>
          <secid>default</secid>
          <default/>
        </section>
      </disclose>
    </subscribe>

    <command>
      <name>FETCH</name>
      <allow>
        <group>group1</group>
        <group>group2</group>
      </allow>
    </command>
    <command>
      <name>SEND</name>
```

```
        <allow>
          <all/>
        </allow>
        <deny>
          <user>pepp://foo.net/unknown</user>
        </deny>
      </command>
    </acl>
```

12. Sample Transcripts

Consider Alice starts to connect to her home server and her PePP address is "pepp://pepp.fujitsu.com/alice".

12.1. LOGIN

```
// In order to login the PePP service, the client initiate the LOGIN
// request to establish a PePP connection.
// Alice connects PePP service,
// opening the PePP connection from her client to port ??? of
// pepp.fujitsu.com

   LOGIN 00153678 PePP/0.5
   From: pepp://pepp.fujitsu.com/alice
   SASL-mechanism: CRAM-MD5
   Auth-State: init


// Response from server.
// A challenge is given as the content

   PePP/0.5 00153678 100 Authentication Continued
   Content-Type: text/plain
   Content-Length: xxx

   1234.14782225@pepp.org


// The client returns a response in the succeeding LOGIN request

   LOGIN 00153679 PePP/0.5
   From: pepp://pepp.fujitsu.com/alice
   Auth-State: continue
   Content-Type: text/plain
   Content-Length: xxx
```

```
      alice b913a602c7eda7a495b4e6e7334d3890


   // Authentication succeeded

      PePP/0.5 00153679 200 OK
```

12.2. CREATESECTION

```
   // Alice creates a new section containing an IM presence.
   // This content is set as a permanent value of this section.

      CREATESECTION pepp://pepp.fujitsu.com/alice 00153783 PePP/0.5
      From: pepp://pepp.fujitsu.com/alice
      Section-ID: PublicIM
      Section-Name: IM
      Content-Type: text/xml
      Content-Length: yyy

      <section>
        <display-name>Instant Message</display-name>
        <communication>
          <status><closed/></status>
          <address>pepp://pepp.fujitsu.com/alice/iibox</address>
        </communication>
      </section>

   // The new section was successfully created.

      PePP/0.5 00153783 202 Section Created
```

12.3. CHANGE

```
   // Alice changes her presence information

      CHANGE pepp://pepp.fujitsu.com/alice 00153793 PePP/0.5
      From: pepp://pepp.fujitsu.com/alice
      Section-ID: PublicIM
      Duration: 180
      Content-Type: text/xml
      Content-Length: xxx

      <section>
        <display-name>Instant Message</display-name>
        <note>Meeting, Room 5B</note>
        <communication>
```

```
          <status><online><away/></online></status>
          <address>pepp://pepp.org/alice/iibox</address>
        </communication>
      </section>

   // Presence information was successfully changed

      PePP/0.5 00153793 200 OK
      Duration: 180
```

12.4. SUBSCRIBE

```
   //  Bob subscribes to Alice's presence information.
   //  Assumes that ACL specifies to show "PublicIM" (Section-ID) for
   //  the subscription from Bob. The section has a Section-Name "IM".
   //  The disclosed content of the presence information is returned
   //  as an initial value.

      SUBSCRIBE pepp://pepp.fujitsu.com/alice 02658472 PePP/0.5
      From: pepp://pepp.org/bob
      Subscription-Mode: Notify
      Regarding: value

   // Subscription has succeeded, and Bob's client receives the
   // response with Alice's PI at the moment.

      PePP/0.5 02658472 201 Subscription Created
      Subscription-ID: a1b2c3d4e5f6
      Content-Type: multipart/mixed; boundary="Multipart0123456789"
      Content-Length: xxx

      --Multipart0123456789
      Content-Type: text/xml
      Content-Length: zzz
      Last-Modified: 2000-06-10T09:10:43Z

      <section>
        <display-name>Instant Message</display-name>
        <note>Meeting, Room 5B</note>
        <communication>
          <status><online><away /></online></status>s
          <address>pepp://pepp.org/alice/iibox</address>
        </communication>
      </section>
      --Multipart0123456789
      ( if other section is allowed for Bob to subscribe, they will appear
here.)
      --Multipart0123456789
```

12.5. NOTIFY

```
// When Alice changes her presence, it is notified to the subscriber.

   NOTIFY pepp://pepp.org/bob 31975431 PePP/0.5
   From: pepp://pepp.fujitsu.com/alice
   Section-Name: IM
   Subscription-ID: a1b2c3d4e5f6
   Last-Modified: 2000-06-10T12:03:22Z
   Content-Type: text/xml
   Content-Length: xxx

   <section>
     <display-name>Instant Message</display-name>
     <note>I'm back!</note>
     <communication>
       <status><online></online></status>
       <address>pepp://pepp.org/alice/iibox</address>
     </communication>
   </section>

// The client returns a successful response.

   PePP/0.5 31975431 200 OK
```

12.6. SEND

```
// Bob sends IM to Alice.

   SEND pepp://pepp.fujitsu.com/alice/iibox 01348590 PePP/0.5
   From: pepp://pepp.org/bob
   Message-ID: 200006101523.ZDFN0478//pepp.org/bob
   Conversation-ID: 200006101523.ZDFN0478//pepp.org/bob
   Content-Type:text/plain
   Context-Length: xxx

   Hello!

// The server returns a successful response if the message is
// deliverable.

   PePP/0.5 01348590 200 OK


// Then the server fowards the message to the client connection.

   SEND pepp://pepp.fujitsu.com/alice/iibox 31978216 PePP/0.5
```

```
From: pepp://pepp.org/bob
Message-ID: 200006101523.ZDFN0478//pepp.org/bob
Conversation-ID: 200006101523.ZDFN0478//pepp.org/bob
Content-Type:text/plain
Context-Length: xxx

Hello!
```

```
// The Alice's client returns a successful response.

PePP/0.5 31978216 200 OK
```

12.7. LOGOUT

```
// Alice closes the PePP connection.

LOGOUT 00258674 PePP/0.5
From: pepp://pepp.fujitsu.com/alice
```

```
// The server returns a successful response.

PePP/0.5 00258674 200 OK
```

```
// Alice's client closes the TCP connection.
```

13. Security Considerations

Security considerations are described in Section 4.2.

14. Acknowledgments

Some of the ideas in this document are inspired by private correspondence with John Stracke, eCal Corp., especially for IM transfer and IM conversation. The authors gratefully acknowledge his contributions.

15. References

[Model] M.Day, J.Rosenberg, H.Sugano, "A Model for Presence and Instant

Messaging", RFC 2778, February 2000.
[Reqts] M.Day, S.Aggarwal, G.Mohr, and J.Vincent, "Instant Messaging / Presence Protocol Requirements", RFC 2779, February 2000.
[RelURL] R.Fielding, "Relative Uniform Resource Locators", RFC1808
[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
[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.
[CRAM-MD5] J.Klensin, R.Catoe and P. Krumviede, "IMAP/POP AUTHorize Extension for Simple Challenge/Response", RFC 2195, September 1997.
[TLS] T.Dierks, and C. Allen, "The TLS Protocol Version 1.0", RFC2246, January 1999.
[MIME] N. Freed et al., "Multipurpose Internet Mail Extensions (MIME) Part One" to "Five", RFC 2045-2049, November 1996.
[URI] T. Berners-Lee, R. Fielding, L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC2396, August 1998.
[ISO8601] "Data elements and interchange formats -- Information interchange -- Representation of dates and times", 1988.
[CC/PP] M.Nilsson, J.Hjelm, H.Ohto, "Composit Capabilities/Preference Profiles: Requirements and Architecture", W3C CC/PP Working Group, http://www.w3.org/TR/CCPP-ra/, February, 2000.

16. Authors' Addresses

Hiroyasu Sugano
Fujitsu Laboratories Ltd.
64 Nishiwaki, Ohkubo-cho
Akashi 674-8555
Japan
email: suga@flab.fujitsu.co.jp

Akinori Iwakawa
Fujitsu Laboratories Ltd.
64 Nishiwaki, Ohkubo-cho
Akashi 674-8555
Japan
email: iwakawa@flab.fujitsu.co.jp

Koji Otani
Fujitsu Laboratories Ltd.
64 Nishiwaki, Ohkubo-cho
Akashi 674-8555
Japan
email: sho@flab.fujitsu.co.jp

Takashi Ohno
Fujitsu Laboratories Ltd.
64 Nishiwaki, Ohkubo-cho
Akashi 674-8555
Japan
email: ohno@flab.fujitsu.co.jp

Shingo Fujimoto
Fujitsu Laboratories of America, Inc.
595 Lawrence Expressway
Sunnyvale, CA 94086
USA
email: shingo@fla.fujitsu.com