1. Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To learn the current status of any Internet-Draft, please check the "iid-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

2. Abstract

Online users have demonstrated a desire to know instantly whether another individual is online, to be notified when another individual arrives online, and to send instant messages. These applications currently use independent, non-standard and usually non-interoperable protocols. The goal is to define a standard protocol so that independently written client and server implementations can participate in a global network of "online" users.

This draft outlines requirements for a Presence Information Protocol to support these applications.

Distribution of this document is unlimited. Please send comments to the RVP discussion list at rvp@iastate.edu, which may be joined by sending a message with subject "subscribe" to rvp-request@iastate.edu. Archives of past messages are available. Send the single word "help" to rvp-request@iastate.edu for more information.
3. Contents

1. Status of this Memo........................................1
2. Abstract....................................................................1
3. Contents....................................................................2
4. Introduction.........................................................3
  4.1. Purpose of a Presence Information Protocol............3
  4.2. Desired Characteristics.................................4
  4.3. Definitions..................................................4
5. Detailed Requirements..........................................5
  5.1. Security Requirements..................................5
    5.1.1. Confidentiality..................................5
    5.1.2. Privacy...........................................5
    5.1.3. Authentication...................................6
  5.2. Addressing Requirements................................6
    5.2.1. Naming Format....................................6
    5.2.2. Locating a Server................................7
  5.3. Container Schema Requirements........................7
    5.3.1. User Schema Requirements........................7
    5.3.2. Group Schema Requirements.......................7
  5.4. Other Requirements....................................7
    5.4.1. Control over peer-to-peer messaging..............7
    5.4.2. Nature of Instant Messaging........................8
    5.4.3. Media Types.......................................8
    5.4.4. Location Independence............................8
    5.4.5. One to many or group messaging....................8
6. Goals......................................................................9
  6.1. Scalable....................................................9
  6.2. Flexible....................................................9
  6.3. Efficient..................................................9
7. Protocol Implementation Issues............................10
  7.1. Transport-Protocol Neutral............................10
  7.2. Text-based...............................................10
  7.3. Supports Versioning....................................10
  7.4. Minimal State............................................10
  7.5. Encryption...............................................10
  7.6. Authentication..........................................10
  7.7. Firewalls................................................11
  7.8. Content types............................................11
8. Non-Goals........................................................11
  8.1. Presence Information and Email........................11
  8.2. Presence Information and LDAP........................12
  8.3. Peer-to-peer vs. client-to-server......................12
9. Discussion Group.................................................12
10. REFERENCES.....................................................12
11. Authors’ Addresses.............................................12
4. Introduction

4.1. Purpose of a Presence Information Protocol

Users want to know instantly when a friend or colleague logs on, and to be able to send this acquaintance a quick message. The desired functionality includes both the ability to subscribe to instant notifications of remote events, such as a friend logging on, and the ability to send instant messages to others.

Also note that in early discussions on the problem there has been agreement that a client-only solution is not feasible. Users wish to be notified when another user comes online. When the target user is offline, there must be another entity responsible for taking subscription requests. The protocol must specify how:

@ a client requests or subscribes to dynamic user status information,

@ a server responds to queries and subscriptions,

@ a client sends instant messages, and

@ a server forwards instant messages to recipients.

Besides being able to subscribe to online status, users may also be able to subscribe to other properties of users or properties of non-user objects.

Users and administrators must be able to set access control levels on queryable objects and properties.

Interest-based messaging, in which a user sends a message to a group of individuals which share some characteristics, should be supported by the protocol. This group of individuals or aggregate entity may be handled by the server much like email distribution lists.

Examples of existing presence information systems include:

@ ActiverseÂ’s ÂDing!Â

@ AOLÂ’s ÂInstant MessengerÂ

@ IChatÂ’s ÂICChat PagerÂ

@ Flash CommunicationsÂ’s ÂFlash CommunicatorÂ

@ MirabilisÂ’s ÂICQÂ

@ MSNÂ’s ÂFriends OnlineÂ

@ PeopleLinkÂ’s ÂPeople LinkÂ
4.2. Desired Characteristics

This section briefly outlines the desired characteristics of a presence information system. Requirements and goals of the protocol are discussed in more detail in sections 5 and 6.

Protocol Characteristics:

1. Can scale to support number of users on internet within foreseeable future
2. Provides rich extensibility
3. Globally unique human-readable address can be assigned to a user without consulting a central authority
4. Server is not required to store messages long-term

User Features:

5. Standard way to find the address of a user
6. Standard way to see if a user is online
7. Supports 1-to-1, 1-to-many and interest-classified messaging
8. Supports acknowledgements of received instant messages
9. Supports a subscription and notification model
10. Responsive: quick notifications of status changes
11. Provides maximum flexibility for message content
12. Supports media negotiation and session initiation: users can switch to another application (server-based or peer-to-peer)
13. The target of subscriptions can list who initiated those subscriptions
14. Supports non-humans on either end (or both) of a subscription, query or message
15. Supports location independence

Security Features:

16. Supports reverse-lookup to find out who has asked for information
17. Provides endpoint to endpoint security
18. Support firewall-friendly deployment

19. Supports access control, at minimum who can read or write to any object, object property, or list of objects.

20. Allows users to maintain privacy by blocking others

4.3. Definitions

This specification uses a number of terms to refer to the roles of participants and the kinds of messages passed between them.

Client: An application program that establishes connections to the server for the purpose of registering to be online or
requesting information on another client or group. Clients may or may not interact directly with a human user.

Server: A program that accepts connections from clients and that is responsible for maintaining online/offline state.

Request: A request is a subscription to a property or a query for a property value, or the setting of the property.

Response: A response is a message from a server to a client who has made a query or who has an active subscription.

Subscription: A subscription is a request from a client to be notified when a property changes or to receive all instant messages on a channel. Commonly, a subscription from user A to user B will be for the purpose of notifying user A when user B’s status changes to “online”.

Property: Presence information and other information should accessed as properties, or name/value pairs.

Property Container: A property container can represent a real-world object which has properties queryable through the Presence Information Protocol.

5. Detailed Requirements

5.1. Security Requirements

5.1.1. Confidentiality

Users MUST be able to protect information about themselves. This requirement should be considered met by a system which allows users to set fine-grained access control levels on client or server held objects and properties.

5.1.2. Privacy

Access Control

Users MUST be able to set access control flexibly for objects and properties of objects. Read and write must be handled separately.

Blocking

Users MUST be able to block all requests and messages from specified parties. When a server is responsible for handling requests or forwarding messages for a user container, the server must know the list of blocked parties and MUST block them.

Users MUST be able to specify blocking lists using either positive or negative blocking lists at their choice. That is, the user MUST be able to state "I do not wish to receive messages
from anyone but the following users and I wish to receive messages from anybody except the following. This requirement is necessary for good user experience with the protocol.

When a party is blocked, the protocol should not require that the party be informed that they are blocked. Instead, the blocking user could appear always offline to the blocked user.

Encryption

The content components of requests and responses MUST be capable of being encrypted.

The protocol should support some mechanism to prevent an active attacker from modifying or replaying messages and to prevent repudiation.

5.1.3 Authentication

Clients should be able to issue authentication for their user and should be able to verify credentials offered by other users. Servers should likewise be able to generate and verify the credentials of users and other servers.

Rather than mandate a particular security authority or mandate signing or sealing in any form, the successful proposal for a protocol should allow for optional inclusion of credentials, keys, signatures and encryption.

5.2 Addressing Requirements

Property containers and properties MUST be identifiable by one or more unambiguous names. The namespace for containers and properties should be federated (distributed across the network and with no central management structure).

URI naming would meet the above stated requirements.

There MUST NOT be a requirement for a central address-management server.

User addresses SHOULD be closely linked to a user's name. This may require addresses to be represented in Unicode in order to represent names which contain non-Latin characters.

5.2.1 Naming Format

There should be a naming format usable for this system. This should be flexible enough so that the user entering a name into a client can:

- Specify a property or a node
- Specify whether to query or subscribe to a property

Calsyn & Dusseault
5.2.2. Locating a Server

A name representing a property container or property in the PIP namespace should be mappable to one or more servers capable of answering requests targeted at that container or property.

5.3. Container Schema Requirements

The user container must have some standard properties so that other users know which property to query to find the information most commonly sought. Protocol designers may wish to define schemas for the user container and possibly for other kinds of containers. Schema proposals should be defined separately from the protocol proposals.

5.3.1. User Schema Requirements

The schema for user containers should facilitate, at a minimum:

- A property representing whether a user is online or offline. Other status information (e.g., refining that into ÂbusyÂ, Âdo not disturbÂ, etc) is desirable.
- A writable location for sending instant messages to a recipient
- Access control entries controlling who may view a userÂs status or send message content to the user.

Other properties may be desirable and might become part of a successful draft for a PIP, but the above are sufficient for basic PIP functionality.

5.3.2. Group Schema Requirements

Users should be able to query standard properties of a group to obtain a list of current subscribers. Individual groups may deem this information sensitive and the server may respond that access to this information is denied. A successful PIP must also support control over who may join a group and/or send messages to the group.

5.4. Other Requirements

5.4.1. Control over peer-to-peer messaging

Many clients will wish to exchange messages peer-to-peer. While there are security implications, this can greatly reduce load on servers. The protocol should allow implementations that support this kind of message exchange. The protocol should also allow implementations that do not do this kind of message exchange.
5.4.2.  Nature of Instant Messaging

In order for instant messaging to be real-time, the protocol must not require clients to poll the server. Servers should be able to notify the clients of instant messages.

5.4.3.  Media Types

Content Types

The PIP should support many different types of content in instant messages, such as images, sounds and formatted text.

Languages and Character Sets

The IETF mandates certain levels of internationalization in protocols.

Streaming Media

It is not required for the protocol to support streaming media. In general, instant messages should be brief and self-contained. However, the protocol must be able to support invitation to a streaming-media interaction.

Media Negotiation

Media Negotiation is the ability for clients to transmit information on supported applications or protocols and use this information to choose to enter a session together. This could be a H.323-based or IRC-based session, peer-to-peer or client-to-server, etc.

Users MUST be able to initiate sessions over other protocols. Mandatory client-side support for SIP could be considered a possible solution to this problem. This is an essential usability feature and improves scalability/efficiency of the presence information system by offloading some communication onto other bands.

5.4.4.  Location Independence

User may roam, using different clients on different machines in different locations. However, it is desirable that users remain identifiable with a persistent address, even when roaming.

The protocol MUST provide a mechanism such that a client can access any machine on the network and be able to have messages directed, on a temporary or permanent basis, from their normal machine to this machine. This mechanism could be referral, forwarding, proxying, or some combination of these or other methods.

5.4.5.  One to many or group messaging
The protocol must support the concept of groups or distribution lists for one-to-many messaging.

There must be a way to find out who is receiving instant messages in a group, although the server may respond that access to that list is denied.

6. Goals

The following issues are of particular relevance to the creation of a presence protocol and thus are listed to help the protocol designers in making design choices.

6.1. Scalable

The protocol designers should keep in mind that it will be desirable to accommodate most of the users on the Internet, plus many non-user objects with dynamic data, in a federated presence information system. While not all users of presence information will be online at all times, many will want to keep a "ghost" presence online even when offline.

In order to reduce load on servers, message duplication should be avoided where possible.

6.2. Flexible

Flexible: the protocol should accommodate the design of presence technology implementations with different goals. For example, the protocol should accommodate the following variations

- a combined client and server
- different authentication schemes
- high-security server vs. low-security server
- additional dynamic properties
- different schemas in addition to a user schema

6.3. Efficient

Efficiency: the protocol SHOULD avoid message duplication when possible. This can be done by allowing servers to single-instance messages to another server which can fan out those messages further.

Allowing peer-to-peer communication (both instant messaging and requests/notifications) and session negotiation can make the protocol more efficient by offloading some communication onto other bands.
7. Protocol Implementation Issues

Although this section begins to stray into implementation details, the authors wish to provide some recommendations on the nature of the protocol. These recommendations are based on early discussions on what kind of protocol would be best suited to the task at hand.

7.1. Transport-Protocol Neutral

The protocol should be able to utilize both UDP and TCP as transport protocols. When TCP is used, users should be marked as "offline" if the connection is dropped. When UDP is used, there must be some way for users to periodically assert their presence online.

7.2. Text-based

The protocol should be text-based. This allows easy implementation in languages such as Tcl and Perl, allows easy debugging, and most importantly, makes the protocol flexible and extensible.

7.3. Supports Versioning

There should be some way of exchanging protocol version information.

7.4. Minimal State

In order to support a large number of users in some scenarios, there should be very little state required by the protocol for the server to maintain.

7.5. Encryption

Signing and sealing (encrypting) of protocol elements should use the prevailing standards and be flexible enough to support as yet unspecified new standards in this area. For instance, presuming that MIME is used to encapsulate content, S/MIME should be supported. SSL should be supported for encapsulating of the entire transport stream.

7.6. Authentication

The protocol should facilitate the transport of credentials using prevailing standards (e.g., MIME delivery of certificate information, negotiation of common authentication providers, etc). It is unclear at this time how much of this exchange should occur in the transport and how much should occur within the content components of a PIP protocol.
7.7. Firewalls

The protocol should take into account the broad presence of firewalls, and the desirability for users to be able to exchange instant messages and subscribe to properties from behind firewalls.

The protocol should allow servers to proxy instant messages, queries, subscriptions and responses in order to facilitate server-side security and firewall security.

7.8. Content types

To accommodate rich and varied content components, the PIP SHOULD support MIME as its content encapsulation format. This allows arbitrary byte-pattern data to be sent, and includes common formats such as plain text and HTML text.

MIME types destined for user clients MUST be transported through servers unaltered. Clients SHOULD support some important MIME types such as text and HTML-formatted text. Multipart-alternative formats should be supported by clients to facilitate the rendering of a single message by clients of differing capabilities.

Clients will need to understand some basic MIME types in order for users to communicate but need not support most or all.

8. Non-Goals

In order to make progress in a reasonable time, the designers of a presence information protocol should avoid taking on problems with too large a scope or problems which have been previously solved. In the interest of preventing repetitive discussions on these topics, they are outlined here.

8.1. Presence Information and Email

It is not a goal of this effort to replace email. While email protocols offer some experience and features we can benefit from when designing a system for instant messaging, we are not interested in using an existing email protocol.

Some points of difference between instant messaging and email:

@ Instant messages cannot wait for the user to poll.

@ Instant messaging should have no requirement for the server to store messages for unreachable or offline users.

@ Users need to be able to differentiate between their asynchronous messages and their synchronous messages. Using the same system would confound the two.
Notifications are meant to overwrite old state information rather than add to it. Whereas an email model might work for instant messages, it would be less applicable to notifications.

### 8.2. Presence Information and LDAP

It is not a goal of this effort to replace directory servers which excel in presenting static data. LDAP also can offer us some experience, ideas and has useful features, but LDAP is not designed for dynamic data or for server-initiated sending of data. LDAP also lacks the instant messaging functionality which we require.

### 8.3. Peer-to-peer vs. client-to-server

It is not a goal of this effort to restrict communication to either peer-to-peer the or the client-to-server model. Both models must be supported. Servers are required in some settings to proxy for users and to respond to requests for information on offline users. Peer-to-peer communication is desirable in many situations to avoid load on servers.

### 9. Discussion Group

A discussion group on Presence Information Protocols is available. The authors welcome your participation in the shaping of this draft and the several proposals for protocols in this space.

You can join the discussion list by sending a message body containing the single word `subscribe` (without the quotes) to rvp-request@iastate.edu. Mail to the list recipients should be sent to rvp@iastate.edu. Archives of past messages are available. Send the single word `help` to rvp-request@iastate.edu for more information.

### 10. REFERENCES


### 11. Authors’ Addresses

Martin R. Calsyn  
Microsoft Corporation  
One Microsoft Way  
Redmond, WA  98052

Email: martinca@microsoft.com

Lisa Dusseault  
Microsoft Corporation