Network Working Group
Request for Comments: 3859
Category: Standards Track

J. Peterson
NeuStar

August 2004

# Common Profile for Presence (CPP)

## Status of this Memo



## Copyright Notice



## Abstract

At the time this document was written, numerous presence protocols were in use (largely as components of commercial instant messaging services), and little interoperability between services based on these protocols has been achieved. This specification defines common semantics and data formats for presence to facilitate the creation of gateways between presence services.

## Table of Contents

## 1. Introduction

Presence is defined in RFC2778 [5]. At the time this document was written, numerous presence protocols are in use (largely as components of commercial instant messaging services), and little interoperability between services based on these protocols has been achieved. This specification defines semantics and data formats for common services of presence to facilitate the creation of gateways between presence services: a common profile for presence (CPP).

Service behavior is described abstractly in terms of operations invoked between the consumer and provider of a service. Accordingly, each presence service must specify how this behavior is mapped onto its own protocol interactions. The choice of strategy is a local matter, providing that there is a clear relation between the abstract behaviors of the service (as specified in this memo) and how it is faithfully realized by a particular presence service. For example, one strategy might transmit presence information as key/value pairs, another might use a compact binary representation, and a third might use nested containers.

The parameters for each operation are defined using an abstract syntax. Although the syntax specifies the range of possible data values, each presence service must specify how well-formed instances of the abstract representation are encoded as a concrete series of bits.

In order to provide a means for the preservation of end-to-end features (especially security) to pass through presence interoperability gateways, this specification also provides recommendations for presence document formats that could be employed by presence protocols.

## 2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [1] and indicate requirement levels for compliant implementations.

This memos makes use of the vocabulary defined in RFC 2778 [5]. Terms such as CLOSED, INSTANT INBOX, PRESENCE, and OPEN are used in the same meaning as defined therein.

The term 'gateway' used in this document denotes a network element responsible for interworking between diverse presence protocols. Although the presence protocols themselves are diverse, under the model in this document these protocols can carry a common payload that is relayed by the gateway. Whether these interworking intermediaries should be called 'gateways' or 'relays' is therefore somewhat debatable; for the purposes of this document, they are called 'CPP gateways'.

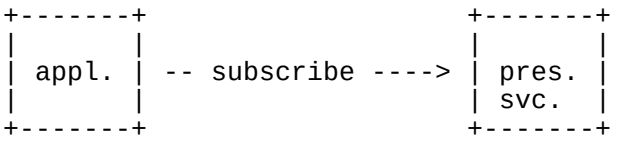
The term 'presence service' also derives from RFC 2778, but its meaning changes slightly due to the existence of gateways in the CPP model. When a client sends an operation to a presence service, that service might either be an endpoint or an intermediary such as a CPP gateway - in fact, the client should not have to be aware which it is addressing, as responses from either will appear the same.

This document defines operations and attributes of an abstract presence protocol. In order for a compliant protocol to interface with a presence gateway, it must support all of the operations described in this document (i.e., the presence protocol must have some message or capability that provides the function described by all given operations). Similarly, the attributes defined for these operations must correspond to information available in the presence protocol in order for the protocol to interface with gateways defined by this specification. Note that these attributes provide only the minimum possible information that needs to be specified for interoperability - the functions in a presence protocol that correspond to the operations described in this document can contain additional information that will not be mapped by CPP.

## 3. Abstract Presence Service

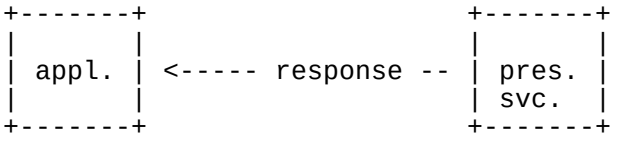
### 3.1. Overview of the Presence Service

When an application wants to subscriber to the presence information associated with a PRESENTITY (in order to receive periodic notifications of presence information), it invokes the subscribe operation, e.g.,



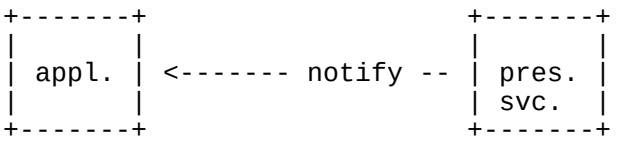

The subscribe operation has the following attributes: watcher, target, duration, SubscriptID and TransID. The 'watcher' and 'target' identify the WATCHER and PRESENTITY, respectively, using the identifiers described in Section 3.2. The duration specifies the maximum number of seconds that the SUBSCRIPTION should be active (which may be zero, in which case this is a one-time request for presence information). The SubscriptID creates a reference to the SUBSCRIPTION that is used when unsubscribing. The TransID is a unique identifier used to correlate the subscribe operation with a response operation. Gateways should be capable of handling TransIDs and SubscriptIDs up to 40 bytes in length.

Upon receiving a subscribe operation, the service immediately responds by invoking the response operation containing the same TransID, e.g.,




The response operation has the following attributes: status, TransID, and duration. 'status' indicates whether the subscribe operation has succeeded or failed. The TransID of the response operation corresponds to the TransID of the subscription operation to which it is responding. The 'duration' attribute specifies the number of seconds for which the subscription will be active (which may differ from the value requested in the subscribe operation).

If the response operation indicates success, the service immediately invokes the notify operation to communicate the presence information to the WATCHER, e.g.,

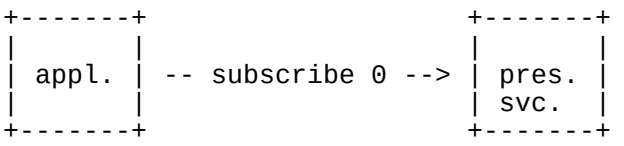



The notify operation has the following attributes: watcher, target, and TransID. The values of 'watcher' and 'target' are identical to those given in the subscribe operation that triggered this notify operation. The TransID is a unique identifier for this notification.

The notify operation also has content, namely PRESENCE INFORMATION. Content details are specified in Section 3.3.

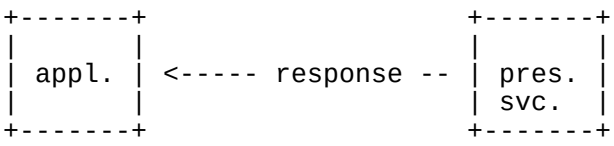
If the duration parameter is non-zero, then for up to the specified duration, the service invokes the notify operation whenever there are any changes to the PRESENTITY's presence information. Otherwise, exactly one notify operation is invoked, achieving a one-time poll of the presence information. Regardless, there is no application response to the notify operation (i.e., the application does not invoke a response operation when a notify operation occurs) defined in CPP.

The application may prematurely cancel a subscription by re-invoking the subscribe operation (as described above) with a duration of 0 and the same SubscriptID as the original subscribe operation , e.g.,




Note that a notify operation will be invoked when a subscription is prematurely canceled in this fashion; this notification may be discarded by the watcher.

The service immediately responds by invoking the response operation containing the same TransID; e.g.,




Note that this specification assumes that CPP-compliant presence protocols provide reliable message delivery; there are no application-layer message delivery assurance provisions in this specification.

## 3.2. Identification of PRESENTITIES and WATCHERS

A PRESENTITY is specified using the PRES URI scheme, which is further described in Appendix A. An example would be: "pres:fred@example.com"

WATCHERs identify themselves in the same manner as PRESENTITIES; that is, with a pres URI.

### 3.2.1. Address Resolution

A presence service client determines the next hop to forward an operation to by resolving the domain name portion of the service destination. Compliant implementations SHOULD follow the guidelines for dereferencing URIs given in [2].

## 3.3. Format of Presence Information

This specification defines an abstract interoperability mechanism for presence protocols; the message content definition given here pertains to semantics rather than syntax. However, some important properties for interoperability can only be provided if a common end-to-end format for presence is employed by the interoperating presence protocols, especially with respect to security. In order to maintain end-to-end security properties, applications that send notification operations through a CPP gateway MUST support the format defined in PIDF [4]. Applications MAY support other content formats.

CPP gateways MUST be capable of relaying the body of a notification operation between supported presence protocols without needing to modify or inspect the content.

### 3.4. The Presence Service

An implementation of the service must maintain information about both presence information and continual operations (like periodic notification) in persistent storage.

Note that the subscription-identifier attribute used by the subscribe operation is potentially long-lived. Accordingly, the values generated for this parameter should be unique across a significant duration of time. The SubscriptID parameter should be intrinsically globally unique over time, not merely unique among operations sent to or from a particular WATCHER and PRESENTITY.

#### 3.4.1. The Subscribe Operation

When an application wants to subscribe to the presence information associated with a PRESENTITY, it invokes the subscribe operation.

When the service is informed of the subscribe operation, it performs these steps:

1. If the watcher or target parameter does not refer to a valid PRESENTITY, a response operation having status "failure" is invoked.

2. If access control does not permit the application to request this operation, a response operation having status "failure" is invoked.

3. If the duration parameter is non-zero, and if the watcher and target parameters refer to an in-progress subscribe operation for the application, a response operation having status "failure" is invoked.

4. Otherwise, if the service is able to successfully deliver the message:

   A response operation having status "success" is immediately invoked. (If the service chooses a different duration for the subscription then it conveys this information in the response operation.)

   A notify operation, corresponding to the target's presence information, is immediately invoked for the watcher.

For up to the amount of time indicated by the duration parameter of the notify operation (measured from the time that the subscribe operation was received), if the target's presence information changes, and if access control allows, a notify operation is invoked for the watcher.

Note that if the duration parameter is zero-valued, then the subscribe operation is making a one-time poll of the presence information. Accordingly, the final step above (continued notifications for the duration of the subscription) does not occur.

When the service invokes a response operation as a result of this processing, the transID parameter is identical to the value found in the subscribe operation invoked by the application.

### 3.4.2. The Notify Operation

The service invokes the notify operation whenever the presence information associated with a PRESENTITY changes and there are subscribers requesting notifications for that PRESENTITY.

There is no application response to the notify operation.

### 3.4.3. Subscribe Operation (with Zero Duration)

When an application wants to terminate a subscription, it issues a SUBSCRIBE 0 with the SubscriptID of an existing subscription. Note that a notify operation will be invoked by the presentity when a subscription is canceled in this fashion; this notification can be discarded by the watcher. There is no independent UNSUBSCRIBE operation.

When an application wants to directly request presence information to be supplied immediately without initiating any persistent subscription, it issues a SUBSCRIBE 0 with a new SubscriptID. There is no independent FETCH operation.

## 4. Security Considerations

Detailed security considerations for presence protocols given in RFC 2779 [6] (in particular, requirements are given in sections 5.1 through 5.3 with some motivating discussion in 8.2).

CPP defines an interoperability function that is employed by gateways between presence protocols. CPP gateways MUST be compliant with the minimum security requirements of the presence protocols with which they interface.

The introduction of gateways to the security model of presence in RFC 2779 also introduces some new risks. End-to-end security properties (especially confidentiality and integrity) between presentities and watchers that interface through a CPP gateway can only be provided if a common presence format (such as the format described in [4]) is supported by the protocols interfacing with the CPP gateway.

When end-to-end security is required, the notify operation MUST use PIDF, and MUST secure the PIDF MIME body with S/MIME [8], with encryption (CMS EnvelopeData) and/or S/MIME signatures (CMS SignedData).

The S/MIME algorithms are set by CMS [9]. The AES [11] algorithm should be preferred, as it is expected that AES best suits the capabilities of many platforms. Implementations MAY use AES as an encryption algorithm, but are REQUIRED to support only the baseline algorithms mandated by S/MIME and CMS.

When PRES URIs are used in presence protocols, they convey the identity of watchers and/or presentities. Certificates that are used for S/MIME presence operations SHOULD, for the purposes of reference integrity, contain a subjectAltName field containing the PRES URI of their subject. Note that such certificates may also contain other identifiers, including those specific to particular presence protocols. In order to further facilitate interoperability of secure presence services through CPP gateways, users and service providers are encouraged to employ trust anchors for certificates that are widely accepted rather than trust anchors specific to any particular presence service or provider.

In some cases, anonymous presence services may be desired. Such a capability is beyond the scope of this specification.

## 5. IANA Considerations

The IANA has assigned the "pres" URI scheme.

### 5.1. The PRES URI Scheme

The Presence (PRES) URI scheme designates an Internet resource, namely a PRESENTITY or WATCHER.

The syntax of a PRES URI is given in Appendix A.

## 6. Contributors

Dave Crocker edited earlier versions of this document.

The following individuals made substantial textual contributions to this document:

Athanassios Diacakis (thanos.diacakis@openwave.com)

Florencio Mazzoldi (flo@networkprojects.com)

Christian Huitema (huitema@microsoft.com)

Graham Klyne (gk@ninebynine.org)

Jonathan Rosenberg (jdrosen@dynamicsoft.com)

Robert Sparks (rsparks@dynamicsoft.com)

Hiroyasu Sugano (suga@flab.fujitsu.co.jp)

## 7. References

### 7.1. Normative References

[1] Bradner, S., "Key words for use in RFCs to indicate requirement levels", BCP 14, RFC 2119, March 1997.

[2] Peterson, J., "Address Resolution for Instant Messaging and Presence", RFC 3861, August 2004.

[3] Resnick, P., "Internet Message Format", STD 11, RFC 2822, April 2001.

[4] Sugano, H., Fujimoto, S., Klyne, G., Bateman, A., Carr, W., and J. Peterson, "Presence Information Data Format (PIDF)", RFC 3863, August 2004.

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

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

[7] Allocchio, C., "GSTN Address Element Extensions in Email Services", RFC 2846, June 2000.

[8] Ramsdell, B., "Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.1 Message Specification", RFC 3851, July 2004.

[9] Housley, R., "Cryptographic Message Syntax (CMS)", RFC 3852, July 2004.

[10] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

## 7.2. Informative References

[11] Schaad, J., "Use of the Advanced Encryption Standard (AES) Encryption Algorithm and in Cryptographic Message Syntax (CMS)", RFC 3565, July 2003.

## Appendix A. PRES URI IANA Registration Template

This section provides the information to register the pres: presence URI .

### A.1. URI Scheme Name

pres

### A.2. URI Scheme Syntax

The syntax follows the existing mailto: URI syntax specified in RFC 2368. The ABNF is:

```
PRES-URI        = "pres:" [ to ] [ headers ]
to              =  mailbox
headers         =  "?" header *( "&" header )
header          =  hname "=" hvalue
hname           =  *uric
hvalue          =  *uric
```

Here the symbol "mailbox" represents an encoded mailbox name as defined in RFC 2822 [3], and the symbol "uric" denotes any character that is valid in a URL (defined in RFC 2396 [10]).

### A.3. Character Encoding Considerations

Representation of non-ASCII character sets in local-part strings is limited to the standard methods provided as extensions to RFC 2822 [3].

### A.4. Intended Usage

Use of the pres: URI follows closely usage of the mailto: URI. That is, invocation of an PRES URI will cause the user's instant messaging application to start, with destination address and message headers fill-in according to the information supplied in the URI.

### A.5. Applications and/or Protocols which use this URI Scheme Name

It is anticipated that protocols compliant with RFC 2779, and meeting the interoperability requirements specified here, will make use of this URI scheme name.

### A.6. Interoperability Considerations

The underlying exchange protocol used to send an instant message may vary from service to service. Therefore complete, Internet-scale interoperability cannot be guaranteed. However, a service conforming to this specification permits gateways to achieve interoperability sufficient to the requirements of RFC 2779.

### A.7. Security Considerations

See Section 4.

### A.8. Relevant Publications

RFC 2779, RFC 2778

### A.9. Person & Email Address to Contact for Further Information

Jon Peterson [mailto:jon.peterson@neustar.biz]

### A.10. Author/Change Controller

This scheme is registered under the IETF tree. As such, IETF maintains change control.

### A.11. Applications and/or Protocols which use this URI Scheme Name

Instant messaging service; presence service

## Appendix B. Issues of Interest

This appendix briefly discusses issues that may be of interest when designing an interoperation gateway.

### B.1. Address Mapping

When mapping the service described in this memo, mappings that place special information into the im: address local-part MUST use the meta-syntax defined in RFC2846 [7].

### B.2. Source-Route Mapping

The easiest mapping technique is a form of source-routing and usually is the least friendly to humans having to type the string. Source-routing also has a history of operational problems.

Use of source-routing for exchanges between different services is by a transformation that places the entire, original address string into the im: address local part and names the gateway in the domain part.

For example, if the destination INSTANT INBOX is "pepp://example.com/fred", then, after performing the necessary character conversions, the resulting mapping is:

im:pepp=example.com/fred@relay-domain

where "relay-domain" is derived from local configuration information.

Experience shows that it is vastly preferable to hide this mapping from end-users - if possible, the underlying software should perform the mapping automatically.

## Appendix C. Acknowledgments

The author would like to acknowledge John Ramsdell for his comments, suggestions and enthusiasm. Thanks to Derek Atkins for editorial fixes.

## Author's Address

Jon Peterson
NeuStar, Inc.
1800 Sutter St
Suite 570
Concord, CA 94520
US

Phone: +1 925/363-8720
EMail: jon.peterson@neustar.biz

## Full Copyright Statement



## Intellectual Property



## Acknowledgement

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