Abstract

In any presence application, it is frequently necessary for the user to configure a number of pieces of information. Users will need to manipulate their presentity list, adding and removing presentities, and manipulate their authorization lists, which specify the set of users that can subscribe to their presence. In this document, we provide a framework and requirements for such data manipulations.
Table of Contents

1 Introduction ........................................ 3
2 Terminology ......................................... 3
3 Framework .......................................... 4
4 Presentity Collection Manipulation Requirements .... 6
5 Authorization Policy Manipulation .................... 9
5.1 Acceptance Policy Requirements ................... 9
5.2 Notification Requirements ........................ 10
5.3 Content Requirements ............................ 10
5.4 General Requirements ............................. 11
6 Security Considerations ............................. 12
7 To Do ............................................... 12
8 Acknowledgements ................................... 13
9 Authors Addresses .................................. 13
10 Normative References ............................... 13
11 Informative References ............................. 13
1 Introduction

Consumer-based instant messaging and presence applications typically provide a rich set of features. In addition to being able to subscribe to, and get notified of, changes in presence, users can also configure the operation of the application.

Most systems allow the user to add or remove users from their "buddy list", which we refer to here as a presentity collection. The presentity collection is the set of presentities [1] that a user is subscribed to. This list is frequently stored on the server, allowing the user to generate a single subscription to the entire list. The server then "fans out" that subscription too all the presentities on the list. Subscription to presentity collections is supported through the SIP event notification extension for collections [2]. However, no automated means is currently defined to create these lists, add users to them, remove users from them, or query for the set of users on the list.

Similarly, most systems support user-defined authorization policies. A user can specify which watchers are (or are not) allowed to subscribe to their presence, and furthermore, what aspects of their presence a watcher is able to see. While SIMPLE [3] systems can support such authorization policies, besides human-driven techniques, such as web or voice response, there is no automated way to specify these policies.

In this document, we propose a framework and a set of requirements for manipulation of presentity collections and authorization policies.

2 Terminology

This document uses the following terminology:

Presentity Collection: A presentity collection is a set of presentities, each of which is identified by a URI. The collection itself is identified by a URI (for example, sip:myfriends@example.com). Using the SIP event extension for collections [2], a watcher can subscribe to the presentity collection and learn about the presence state of all the presentities in the set.

Presence Authorization Policy: Presence authorization policy refers to the set of directives given to a presence agent on what subscriptions to accept, when to generate notifications for a subscription, and what information should be placed in those notifications.
Acceptance Policy: The component of presence authorization policy that determines whether or not to accept a subscription from a watcher.

Notification Policy: The component of presence authorization policy that determines when a notification should be sent to a watcher.

Content Policy: The component of presence authorization that determines the content of the information provided to a watcher in a notification.

SIMPLE Data Elements: SIMPLE data elements are user specified data that determine the behavior of a presence agent. This includes presentity collections and presence authorization policy.

Data Manipulation Client: A data manipulation client is a protocol agent that reads, writes, and receives notifications of changes in SIMPLE data elements.

Data Manipulation Server: A data manipulation server is a protocol agent that receives reads, writes, and sends notifications of changes in SIMPLE data elements. The server is responsible for the storage of the SIMPLE data elements.

3 Framework

The framework for the usage and manipulation of SIMPLE data elements is shown in Figure 1.

The data manipulation client (just referred to as the client) uses some protocol, whose requirements are specified here, to interact with the data manipulation server. Those interactions include requests to read a SIMPLE data element, write one, or receive notifications in changes to one. The data manipulation server (just referred to as the server) manages a persistent store of the SIMPLE data elements, and interacts with the client.

When a Presence Agent (PA) receives a SIP SUBSCRIBE request [3], it may require access to SIMPLE data elements in order to process the request. For example, if the subscription is for a presentity collection, the PA will need to determine that this is the case, and secondly, "expand" the collection, obtaining the list of URIs for that collection.
If the SUBSCRIBE request is for a presentity, the PA will need to obtain the presence authorization policy of that presentity in order to process the SUBSCRIBE request.

In both cases, the PA requires only read access to the data. As a result, it obtains it directly from the data store, rather than interacting with the server. This, of course, is just a model of the system; a real implementation might involve interaction with the...
server before reading the data.

Between the presentity collection and presence authorization policy, the presence authorization policy is a far more complicated piece of data. The authorization policy can be reasonably split into three separate pieces. The first, which we call the acceptance policy, determines whether or not to grant a subscription to the subscriber. This policy results in a binary decision. The second piece, which we call the notification policy, determines when that particular subscriber should receive notifications. For example, a subscriber might only be permitted to see when I log in or log out of IM, but not receive notifications when my phone goes on hook. This is closely related to the third piece, which we call the content policy. This policy specifies the content of the information present in a notification that is sent to a subscriber.

All of these policies are data that is manipulated by the data manipulation protocol.

4 Presentity Collection Manipulation Requirements

The following are the set of requirements for the protocol between the client and the server for the purposes of manipulation presentity collections.

REQ PC-1: It MUST be possible for the client to create a presentity collection and associate it with a URI.

REQ PC-2: It MUST be possible for the user to specify the URI for the presentity collection when one is created. If the name cannot be allocated (because it already exists, for example), it MUST be possible to inform the client of the failure, and the reason for it.

REQ PC-3: It MUST be possible for the server to provide the client a URI for the list when one is created, in the case where the client does not provide it.

REQ PC-4: It MUST be possible to add an entry to the presentity collection. Each entry MUST consist of at least a URI, and MAY include a display name. It MUST be possible for the entry to be any URI that is meaningful in the context of a presentity collection. Examples would include a SIP URI or pres URI [4].

REQ PC-5: It MUST be possible for a presentity collection to contain entries which are themselves presentity collections. It MUST be possible for the client to
determine whether the entry is a presentity or another presentity collection.

REQ PC-6: It MUST be possible to remove an entry from the presentity collection. If the entry does not exist, it MUST be possible for the server to inform the client of this fact.

REQ PC-7: It MUST be possible to clear all entries from a presentity collection.

REQ PC-8: It MUST be possible to delete a presentity collection. In this context, deleted means that the name of the presentity collection is no longer defined, so that subscriptions to the list would fail.

REQ PC-9: It MUST be possible to query for the set of URIs in a particular presentity collection, by providing the URI for the presentity collection.

REQ PC-10: It MUST be possible for the presentity collection to be associated with a list of authorized users. Those authorized users are the only ones permitted to manipulate the presentity collection.

REQ PC-11: It MUST be possible for the presentity collection to be associated with a list of users that are authorized to subscribe to the list.

REQ PC-12: It MUST be possible for a client to store a cached copy of the list. This implies that it MUST be possible for the server to notify the client of a change in the list. It MUST be possible for the client to manipulate the local cached copy even when there is no connectivity to the server. It MUST be possible to synchronize the cached copy with the master copy on the server, when connectivity is re-established.

This particular requirement is crucial for wireless systems, where a copy of the list resides on the handset. Without this requirement, a user would not be able to view the list, or add a user to it, when they go out of coverage.

REQ PC-13: It MUST be possible for there to be multiple clients with cached copies of the list.

REQ PC-14: Manipulations of the presentity collection MUST
exhibit the ACID property; that is, they MUST be atomic, be consistent, durable, and operate independently.

REQ PC-15: It MAY be possible for the client to batch multiple operations (add a presentity, remove a presentity) into a single request that is processed atomically.

REQ PC-16: It MUST be possible for the server to authenticate the client.

REQ PC-17: It MUST be possible to use the same database of client credentials used with SIP and SIMPLE, with the data manipulation protocol.

REQ PC-18: It MUST be possible for the client to authenticate the server.

REQ PC-19: It MUST be possible for message integrity to be insured between the client and the server.

REQ PC-20: It MUST be possible for confidentiality to be ensured between the client and server. As a motivating example, an eavesdropper on the protocol could ascertain the set of people in my presentity collection, resulting in divulging private information.

REQ PC-21: It MUST be possible for the protocol to operate through an intermediary, such as a proxy.

REQ PC-22: It MUST be possible to modify an entry in the presentity collection.

REQ PC-23: It MUST be possible for the protocol to operate with devices with intermittent and low bandwidth connectivity, such as wireless data devices.

REQ PC-24: It MUST be possible for the presence collection to be integrated with a network resident address book. This means that there should be no duplication of data between the two, and only a single transaction should be needed to add or remove an entry from both.

REQ PC-25: It MUST be possible for a user to retrieve the list of collections that they have created.

REQ PC-26: It MUST be possible to associate a display name with a collection.
REQ PC-27: It MUST be possible to extend the set of information associated with entries in the collection.

5 Authorization Policy Manipulation

The following are the set of requirements for the protocol between the client and the server for the purposes of manipulating presence authorization policy. The requirements are divided between acceptance policy, notification policy, and content policy.

5.1 Acceptance Policy Requirements

REQ AP-1: It MUST be possible for the acceptance policy to support rejection of the subscription if the watcher is present on a specified list of "blocked watchers". When a list is checked in this fashion, it's referred to as a blocked list.

REQ AP-2: It MUST be possible for the acceptance policy to support rejection of the subscription if the watcher is not present on a specified list of "allowed watchers".

REQ AP-3: It MUST be possible for the acceptance policy to support making a subscription pending if the watcher is present on neither an explicit allowed or blocked list. In that case, the watcherinfo package [5] can be used for reactive authorization.

REQ AP-4: It MUST be possible for the acceptance policy to check multiple blocked and allowed lists.

REQ AP-5: It SHOULD be possible for the policy to be based on the means by which the authenticated identity of the watcher was determined (digest vs. s/mime, for example).

REQ AP-6: It SHOULD be possible for the policy to be based on whether notifications can be sent encrypted to the subscriber.

REQ AP-7: It MUST be possible for a subscription to be accepted or rejected based on whether the subscriber is on the presentity’s own buddy list.

REQ AP-8: It MUST be possible for the user to manipulate any lists that are checked by the authorization policy (for example, the allowed and denied lists). Manipulate means to add, remove, modify, read, clear and create and delete.
REQ AP-9: It MUST be possible for the acceptance policies to be applied to subscriptions for SIP event packages [6] besides presence.

5.2 Notification Requirements

REQ N-1: It MUST be possible for the user to specify that notifications are to be sent only when the value of a particular status type changes.

REQ N-2: It MUST be possible for the user to specify that the notifications are to be sent only when a particular status type changes to a specified value or set of values.

REQ N-3: It MUST be possible for the user to specify that the notifications are to be sent only when a particular status type changes from a specified value to a specified value (i.e., from open to closed).

REQ N-4: It MUST be possible for the user to specify that the notifications are to be sent only when the value of the contact address changes.

REQ N-5: It SHOULD be possible for the user to specify that the notifications are not, or should be sent on changes in the state of the subscription (as opposed to the state of the presentity).

5.3 Content Requirements

REQ C-1: It MUST be possible for the user to specify that the notification should or should not contain a contact address.

REQ C-2: It MUST be possible for the user to specify that the notification should contain only specific status types (such as basic).

REQ C-3: The user MUST be able to specify the specific values of a specific status type that the notification should or should not contain. Values not permitted must result in the omission of that status type. If all status is omitted, the tuple must be omitted as well. As an example, a user can specify that the notification should include tuples with OPEN status, but suppress those with only CLOSED status.

REQ C-4: The user MUST be able to specify that the notification should only contain information for particular tuples.
REQ C-5: It SHOULD be possible for the user to specify that the value of a status should be modified for a particular subscriber (i.e., the user wants to lie).

REQ C-6: It SHOULD be possible for the user to specify the specific presence document to send to a watcher.

REQ C-7: It SHOULD be possible for the user to specify that the notifications should be encrypted using S/MIME.

5.4 General Requirements

These requirements apply to all of the three components of the authorization policy.

REQ G-1: It MUST be possible for a client to store a cached copy of the policies. This implies that it MUST be possible for the server to notify the client of a change in these data. It MUST be possible for the client to manipulate the local cached copy even when there is no connectivity to the server. It MUST be possible to synchronize the cached copy with the master copy on the server, when connectivity is re-established.

REQ G-2: It MUST be possible for there to be multiple clients with cached copies of the data.

REQ G-3: Manipulations of the data MUST exhibit the ACID property; that is, they MUST be atomic, be consistent, durable, and operate independently.

REQ G-4: It MAY be possible for the client to batch multiple operations (add a user to a list, set a display name) into a single request that is processed atomically.

REQ G-5: It MUST be possible for the server to authenticate the client.

REQ G-6: It MUST be possible to use the same database of client credentials used with SIP and SIMPLE, with the data manipulation protocol.

REQ G-7: It MUST be possible for the client to authenticate the server.

REQ G-8: It MUST be possible for message integrity to be ensured between the client and the server.
REQ G-9: It MUST be possible for confidentiality to be ensured between the client and server. As a motivating example, an eavesdropper on the protocol could ascertain the set of people in my allowed list collection, resulting in divulging private information.

REQ G-10: It MUST be possible for the protocol to operate with devices with intermittent and low bandwidth connectivity, such as wireless data devices.

REQ G-11: It MUST be possible to extend the authorization policies with new rules.

REQ G-12: It MUST be possible for a client to discover the types of authorization policies the server can handle.

6 Security Considerations

There are many security considerations associated with the protocol whose requirements are defined here.

The protocol is used to manipulate data that has a significant impact on the operation of a service provided to a user. In particular, if the data is manipulated by an attacker, the attacker can:

- convey information to subscribers that the presentity wishes to keep private;
- launch denial of service attacks by flooding a subscriber with more presence information than they expected;
- deny service to subscribers or to presentities.

To prevent these attacks, the protocol has to ensure than only authorized users can manipulate the data. Requirements for authentication and authorization are defined above.

Information conveyed in the protocol represents sensitive data. It can include the content of presentity collections and lists of blocked users, both of which reveal personal preferences of a user that they do not wish to convey. As a result, it is necessary that the client authenticate the server, to be sure it is passing this information to a trusted entity. It is also necessary for the protocol to provide encryption services, so that eavesdroppers cannot inspect the data as it passes by.

7 To Do
Align this with the ongoing filter work

Make sure the requirements are consistent with the final protocol mechanism.

8 Acknowledgements

The authors would like to thank Paul Kyzivat for his input.

9 Authors Addresses

Jonathan Rosenberg
dynamicsoft
72 Eagle Rock Avenue
First Floor
East Hanover, NJ 07936
email: jdrosen@dynamicsoft.com

Markus Isomaki
Nokia
Nokia House
Keilalahti, Espoo
Finland
email: markus.isomaki@nokia.com

10 Normative References

11 Informative References


Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

Full Copyright Statement

Copyright (c) The Internet Society (2003). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.
The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.