

A Presence Service for the Venice Service Grid

Markus Hillenbrand, Aneta Kabzeva, Paul Müller
University of Kaiserslautern
D-67653 Kaiserslautern, Germany
{hillenbr, kabzeva, pmueller}@informatik.uni-kl.de

1 Motivation

The Venice Service Grid [4] is a framework for building secure and dependable distributed applications. As a lightweight Grid, Venice allows an easy creation, deployment, maintenance and usage of the provided services. This is achieved by abstracting from the underlying hardware and software and virtualizing resources. The framework is based on service oriented architecture (SOA) [3]. All services are implemented as Web services [1].

Venice focuses on the needs of the end-user. The framework provides not only intuitive means for service access but also offers a Presence Service that simplifies the user interactions inside the Grid. Every Venice user (person, service or resource) registered for the Presence Service creates a dynamic profile (presence information) visible to other users. The service allows users to subscribe to each other and be notified about changes in the subscribed presence information. The publication of presence data can be advantageous for many applications. Knowing the status of his contact persons a user can effectively plan his communication. He can see whether the other party is available and which communication means (i.e. telephone or e-mail) is most suitable for contacting the person. Also knowing the presence information of services and resources a subscriber can estimate whether he can efficiently use their functionality (e.g. a printer can tell about its paper and toner status). Additionally with presence data an administrator can even easier maintain the services and resources in his area of accountability.

2 The Venice Presence Service

The Venice Presence Service is based on the abstract model for presence [2] of the IETF IMPP Working Group. In this model the users of a presence service are divided into two groups – *Presentities* and *Watchers*. Presentities provide their presence information to be stored and distributed by the service. Watchers are the users who receive presence

information from the service. These entities are almost always combined in presence implementations – every user interacts with the system both as Presentity and as Watcher. Watchers are themselves separated in two sets – *Subscribers* and *Fetchers*. Subscribers are those Watchers who automatically receive a notification from the service when a profile they are registered for changes. A Fetcher asks the presence service for the presence information of a Presentity whenever he needs it.

The structure of the presence information documents adheres to the Presence Information Data Format (PIDF) [7] of the IETF. Every PIDF presence entry consists of one or more tuples. Each tuple contains a basic availability status, optional contact address, optional timestamp for the last update of the data, and other optional data. The basic status is either open (i.e. user is available) or closed (i.e. user is not available). Being based on XML, PIDF is very extensible and allows for adding additional XML namespaces to the basic structure.

The top-level architecture of the developed Presence Service and the interaction between the various services therein is shown in Figure 1. Eight Venice Web services work together to implement the functionality of the overall Presence Service group. The Presentity and Watcher services define the operations for the user interfaces (i.e. graphical user agents that interact with the services). The Presence Information Manager (PIM) is responsible for registering for and unregistering from the Presence Service. This service also saves the user presence profiles. Every registered Presentity can manage his own presence profile (i.e. he can insert new presence information and change or delete existing information). With the help of PIM every registered Watcher can search for Presentities. If a Watcher wants to see the profile of a Presentity he has to subscribe for it. An Open Subscriptions Manager (OSM) service stores a subscription request and passes it on to the Presentity. The Presentity has the opportunity to confirm or decline the request. Declined subscriptions are managed by the Denied Subscriptions Manager (DSM). DSM informs the Watcher that the desired subscription was denied and deletes the re-

quest. Confirmed subscription requests are processed by the Confirmed Subscriptions Manager (CSM). After receiving the confirmation from the Presentity, CSM instructs the Watcher Manager to store the user name and type (i.e. Subscriber or Fetcher) of the Watcher in the Watcher List of the Presentity. If the Watcher is a Subscriber CSM also orders the PIM to send the desired presence information to him. Before reaching the Subscriber the information can be filtered by the Filter Manager (FM). FM allows Subscribers to define filters on their subscriptions according to [5, 6]. Thus they can specify the preferred information to be delivered by notification and when it is to be delivered (i.e. what kind of changes in a profile trigger a notification). However, in the current version of the Presence Service the FM is not yet implemented.

Storing the Watcher List in the Watcher Manager allows every user to manage his Watchers. Thus every Presentity is able to cancel a subscription for his presence information at any time. Watchers are also given the opportunity to terminate a requested or confirmed subscription at any time. They can query this information from the OSM and CSM services, respectively.

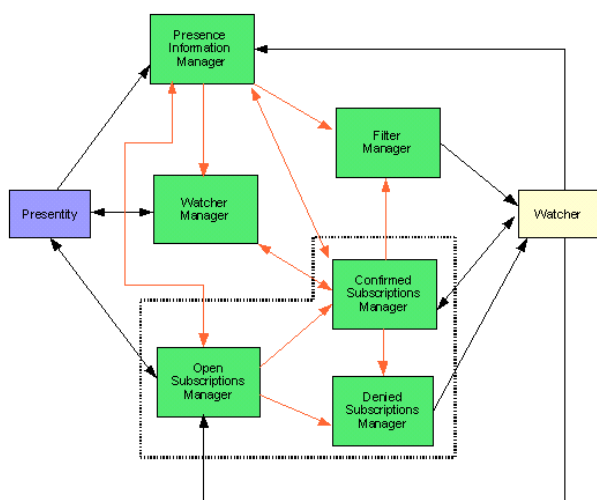


Figure 1. Architectural Layout of the Venice Presence Service

3 Conclusion and Future Work

The Venice Presence Service allows every Venice user (person, service or resource) to show his availability status to others and to see the status of others when the appropriate permissions are given. This information helps users to better plan their communication and to use the functionalities of services and resources more efficiently. Better maintenance of services and resources is also supported.

The present version of the service allows managing the own profile, watcher list and subscriptions. It is possible to search for Presentities and subscribe for presence information.

In the future the implementation of the Filter Manager will allow subscribers to define when to be notified and which information they prefer to be included in the notification. This will reduce not only the size but also the number of messages that have to be sent between the several Web services of the Venice Presence Service. Thus the scalability of the Presence Service will be positively influenced.

4 Acknowledgments

This work has been funded by SIEMENS AG, Munich.

References

- [1] Gustavo Alonso, Fabio Casati, Harumi Kuno, and Vijay Machiraju. *Web Services – Concepts, Architectures and Applications*. Springer Verlag, 2004.
- [2] M. Day, J. Rosenberg, and H. Sugano. RFC 2778 – A Model for Presence and Instant Messaging, 2 2000. <http://www.ietf.org/rfc/rfc2778.txt>.
- [3] Thomas Erl. *Service-Oriented Architecture – Concepts, Technology, and Design*. Prentice Hall, 2005.
- [4] Markus Hillenbrand, Joachim Götze, Ge Zhang, and Paul Müller. A Lightweight Service Grid based on Web Services and Peer-to-Peer. In *Proceedings of Kommunikation in verteilten Systemen KiVS (Berne, Switzerland)*, 2007.
- [5] H. Khartabil, E. Leppanen, M. Lonnfors, and J. Costa-Requena. RFC 4660 – Functional Description of Event Notification Filtering, 9 2006. <http://www.ietf.org/rfc/rfc4660.txt>.
- [6] H. Khartabil, E. Leppanen, M. Lonnfors, and J. Costa-Requena. RFC 4661 – An Extensible Markup Language (XML)-Based Format for Event Notification Filtering, 9 2006. <http://www.ietf.org/rfc/rfc4661.txt>.
- [7] H. Sugano, S. Fujimoto, G. Klyne, A. Bateman, W. Carr, and J. Peterson. RFC 3863 – Presence-Information Data Format (PIDF), 8 2004. <http://www.ietf.org/rfc/rfc3863.txt>.