Development of Secure Event Service for Ubiquitous Computing*

Younglok Lee[1], Seungyong Lee[1], and Hyunghyo Lee[2,**]

[1] Dept. of Information Security, Chonnam National University, Gwangju, 500-757, Korea
dogu@jnu.ac.kr, birch@lsrc.jnu.ac.kr
[2] Div. of Information and EC, Wonkwang University, Iksan, 570-749, Korea
hlee@wonkwang.ac.kr

Abstract. In ubiquitous computing, application should adapt itself to the environment in accordance with context information. Context manager is able to transfer context information to application by using event service. Existing event services are mainly implemented by using RPC or CORBA. However, since conventional distributed systems concentrate on transparency hiding network intricacies from programmers - treating them as hidden implementation details that the programmer must implicitly be aware of and deal with, it is not easy to develop reliable distributed services. Jini provides some novel solutions to many of the problems that classical systems have focused on, and makes some of the problems that those systems have addressed simply vanish. But there is no event servicein Jini. In this paper, we design and implement a secure event service, SeJES, based on Jini in order to provide reliable ubiquitous environment. By using the proposed event service, event consumers are able to retrieve events based on the content. In addition, it enables only authorized suppliers and consumers to exchange event each other. We use SPKI/SDSI certificates in order to provide authentication and authorization and extend JavaSpaces package in order to provide a content-based event retrieval service.

1 Introduction

In ubiquitous computing environment, application should be able to properly adapt itself according to its own context information coming from ubiquitous sensors. Most of the existing communications are based on *request-reply* communication model. However, many ubiquitous computing applications require more flexible and indirect or asynchronous communication mechanism. Event Service[1] is the one which can be used for these asynchronous communications. By using CORBA Event Service, a number of event suppliers and consumers can asynchronously communicate even with no background knowledge with each other. Suppliers and consumers never directly connect to each other and communicate with each other through the event channel.

* This research was supported by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment).
** Corresponding author.

D.-S. Huang, K. Li, and G.W. Irwin (Eds.): ICIC 2006, LNCIS 344, pp. 83 – 94, 2006.

Generally, the Event Service of CORBA (Common Object Request Broker Architecture) can be really applied to many applications. However, in the case of using the event service of CORBA, problems such as persistency and filtering should be solved. Also since the existing distributed system such as CORBA can not provide a reliable environment to develop distributed service, Jini was emerged to solve the problem. In addition, the several features and services of Jini support the characteristics of ubiquitous computing environment. Among those Jini's services is JavaSpaces[2]. But there is no event service in Jini.

JavaSpaces™ is a networked repository for java objects, which provides methods of sharing and transferring objects even if Jini applications do not have any knowledge with each other. Even though applications can utilize an object transferring functions provided by JavaSpaces in order to asynchronously communicate, there are still two problems in doing it. One is that the remote listener which receives the event from JavaSpaces has to call a *read*() method on the JavaSpaces proxy. The other is that the event acquired by that remote method *read*() is not guaranteed as the latest. By modifying and extending JavaSpaces, we implemented JES (JavaSpaces based Event Service)[3]. But there is no security service in JES

In this paper, we implement *SeJES* (Secure JavaSpaces based Event Service) by extending JES to be more applicable to ubiquitous environment. No matter what degree of computing power event consumers have, the proposed SeJES provides secure communication and content based filtering services. Our SeJES performs basic but essential security services such as authentication and authorization using SPKI/SDSI certificates as well. Also, our SeJES provides an enhanced security by transmitting renewed secure session key to event service consumers and suppliers, and some functions of QoS.

This paper consists of as follows: Section 2 briefly reviews related work, and section 3 explains a system model of the SeJES. Section 3 also briefly describes how each component consisting of the system model can be implemented. Section 4 describes how the SeJES be implemented. In this section, we define interfaces furnished to consumer and suppliers, which use Event service. We explain the implementation scenario applied on our system. Finally, conclusion and the further research work are shown in Section 5.

2 Related Work

Many CORBA vendors have developed Event Services, compliant with the OMG specification. Some have even added their own functionality to overcome the drawbacks depending on the application domain in which the Event Service is to be used. This section describes some commercial and academic event models that are CORBA based, and looks at their advantages and drawbacks[4].

IONA Technologies provides two different types of message service products; OrbixEvents[5] and OrbixTalk[6]. OrbixEvents is a C++ implementation of the OMG CORBA Event Service and the only commercial available Event Service that supports

both, untyped and typed events for the push and pull communication models. OrbixTalk is used for distributing IDL-based operations over UDP using either a simple or reliable multicasting, which is ideal in systems that have many consumers and suppliers, since with UDP there is no need to maintain the connection between each consumer and supplier. However, since it is based on UDP it cannot interoperate with any other ORB system.

jEVENTS[7] is a Java implementation of CORBA Event Service for untyped messages, produced by a company called Outack Resource Group Inc. jEVENT supports both push and pull style communication between suppliers and consumers, is also IIOP compliant and may be used with any ORB that supports IIOP.

The VisiBroker Event Service is available in both, C++ and Java versions from Inprise Corporation. Both versions are compliant with the OMG CORBA Event Service implementing untyped events for push and pull communication models. When used with VisiBroker ORB and its Smart Agent architecture that is vendor specific, it becomes a highly available self-recovering service.

ICL have developed a multicast Event Service called DAIS[8]. A multicast service was developed due to the requirements from a customer that needed to communicate sixty messages per second, each being a few hundred bytes in size, to over eight hundred consumers. With these requirements, a standard CORBA Event Channel would have to produce nearly half a million individual messages per second, clearly unfeasible for a distributed system. To minimize the amount of network traffic between supplier and consumer applications, messages are collected into packets and sent in a single UDP packet.

However, existing distributed systems such as CORBA do not consider transferring delay and system performance as a part of program models. They also treat problems concerning network programming as what a programmer should deal with by himself. Therefore, existing distributed systems have difficulty in providing reliable distributed service. But Jini[11], ubiquitous middleware, provides some novel solutions to many of the problems that classical systems have focused on, and makes some of the problems that those systems have addressed simply vanish.

Jini supports serendipitous interactions among services and users of those services. Jini allows services to come and go without requiring any static configuration or administration. Also Communities of Jini services are largely self-healing[12]. But in Jini, there is no event service.

3 System Model

The system model of the SeJES consisting of four components can be distributed among hosts across the network as shown in figure 3.1. The main function of the SeJES is that only the authorized event consumers and producers can exchange events by verifying their identities and capabilities. Event service registers itself with lookup service, which allows event producers and consumers to get the event service proxy. We implement each component of the model described above as follows:

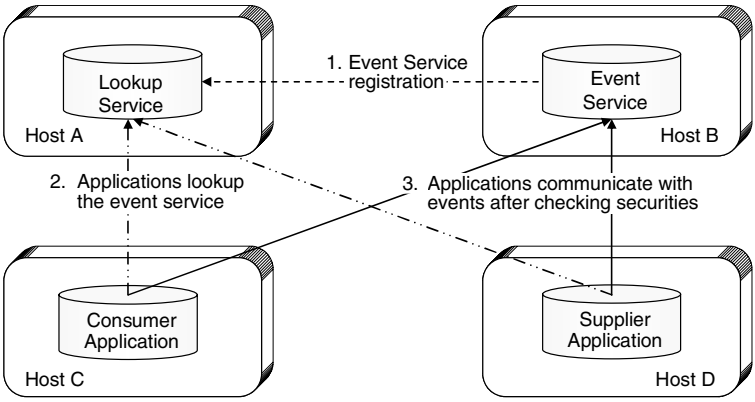


Fig. 3.1. System Model of SeJES

- *Discovery Service* – Event producer and event supplier applications discover the ev ent service by using Lookup Service. We use *reggie*, developed by Sun, which com plies with discovery service specification of Jini.
- *Secure Event Service* – Our SeJES(Secure JavaSpaces based event service) consists of four components(LRC, JES, SSCM, and ERC) as shown in figure 3.2. Major tasks of the LRC (Listener Registration Controller) are not only to retrieve events based on event types and contents, but also register consumers' listener with the JES (JavaSpaces based Event Service) in order for the only authorized consumers to listen events. The JES plays a central role in notifying events provided by event suppliers to its registered consumers and storing the events for content based retrieval. The SSCM(SPKI/SDSI Certificate Manager) proves whether certificate list given from consumer is correct or not, and then returns ACLs (Access Control Lists), related to the events which consumers wish to get, to consumers. The ERC (Event Registration Controller) registers the type of event and ACL, sent by event suppliers, with the SSCM, and checks whether supplied events are authorized or not.

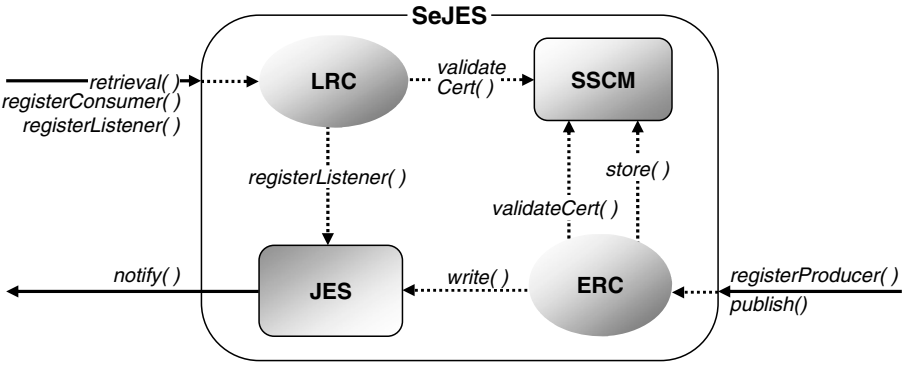


Fig. 3.2. Functional Components of SeJES

- *Event consumer* – Event consumer applications are classified as two types. One is run in machines with powerful computing power and the other does with low cost equipment. In the case of the former, event consumer owns certificate chain discovery algorithm and directly calculates certificate paths which prove that the consumer can get the event, and then delivers them to the SSCM. However, the latter sends all of its name certificates and authorization certificates to the SSCM in order to discover certificate chain lists, which authorize the consumers to get the event. Event consumer applications are able to retrieve with the events as content based after finding event service from discovery service. Furthermore, the consumer is able to get notified of what it wants, as registering remote listener with event service in real time.
- *Event supplier* – Before sending event object instances to the SeJES, event supplier sends a event type and the ACL corresponded with the event type. After getting secure session ID from the SeJES, event supplier inquires SeJES if it is able to send event by using session ID, and then if permitted, it sends the event object instance.

4 The Design and Implementation of SeJES

4.1 Components of SeJES System

This section describes the functions of each component of our SeJES and explains interfaces provided in each module. In addition, we define ACL which event suppliers will provide with and SPKI/SDSI certificates [9] which event consumer will use. Finally, we show the details of SeJES operation by providing the event service usage scenario.

4.1.1 ACL and SPKI/SDSI Certificate

ACL(Access Control List) is a form of expressing security policy that defines which event supplier(issuer) delegates authorization to event consumers(subjects) who will get his events.

< issuer, subject, delegation-bit, authorization tag, validity >

Event consumer is granted the following SPKI/SDSI name and authorization certificates from event supplier.

Name certificate - <issuer, local Name, subject, validity>
Authorization certificate - <issuer, subject, delegation-bit, authorization, validity>

Figure 3.3 explains the S-expression of the authorization certificate that Bob grants its authority “get event 2” to subject called XMan in his local name space, from Nov., 20, 2005 to June, 18, 2006. Name and authorization certificates are sent to event service with their event types when event consumer registers its remote listener with the event service. That is, after finding ACL which fits a

```
(cert   (display plain)
        (issuer  (public-key  (rsa  (e #010001#)
                                    (n |APsREOm+tJQsyS6f7ddzrY4A ...|))))
        (subject  (name XMan))
        (tag “get event 2”)
        (valid    (not-before "2005-11-20_06:51:33")
                  (not-after "2006-6-18_21:51:33"))
        (comment "test certificate"))
(signature   (hash sha1 |aj5Le4mGJ1BltdNdhUm3BVxjgrw=|)
        (public-key (rsa (e #010001#)
                         (n |APsREOm+tJQsyS6f7ddzrY4ACM9fmQC ...|)))
        (rsa-pkcs1-sha1 |mSWhfa2GBJ3YKwkEYL/7yCP3IicwYtCvC ... |))
```

Fig. 3.3. S-expression of authorization certificate

event type by calling SeJES.retrival() on the SeJES proxy, event consumer invoke SeJES.registerConsumer() with the first input “eType” and the second input “certificate-Path”.

4.1.2 LRC

The LRC (Listener Registration Controller) is responsible for registering event listener which event consumer hopes to register with the JES. Using parameter information provided by event consumer, the LRC retrieves ACLs corresponded with the event type and provides methods which can retrieve event based on the event content. It requests authorization check by sending the event type and a bundle of certificates to the SSCM and decides whether it register the event listener or not depending on its returned value.

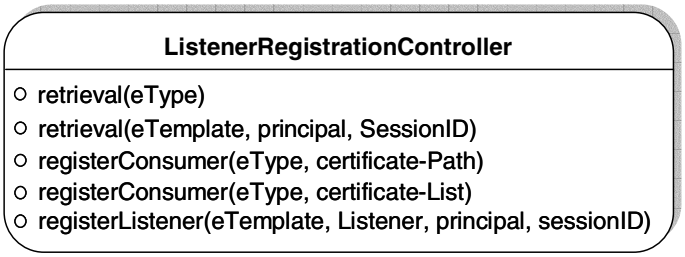
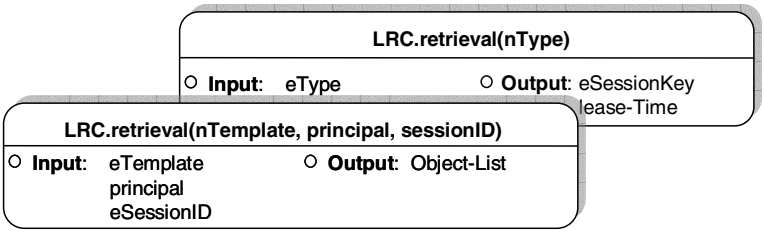


Fig. 3.4. Interfaces of the LRC

- *retrieval(eType), retrieval(eTemplate, principal, sessionID)*

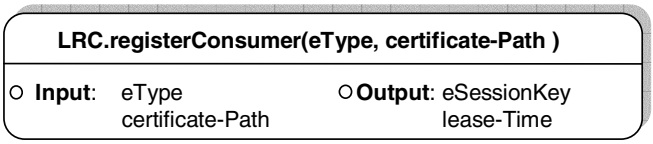
  An Event Consumer invokes the method retrieval(eType) in the LRC proxy so that it can realize if he or she has authorization concerned the event type “eType”. After

finding ACL which represents appropriate authorization related to the event type as a result value, this method returns the ACL to the consumer. In addition, after checking authorization, consumer can invoke a method retrieval(eTemplate, principal, sessionID) of event service proxy in order to retrieve the event based on content. By using three parameters, this method proves if consumer is properly authorized and updates session key as a new value and returns the event which coincides with the eTemplate to the consumer.



- *registerConsumer(eType, cert-Path), registerconsumer(eType, cert-List)*

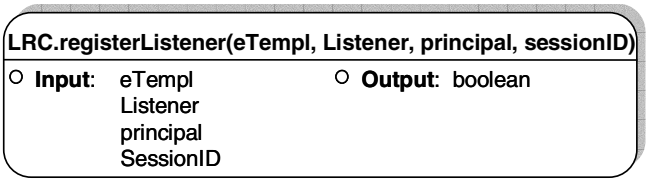
  Event Consumer, running in the machine with computing power, calls a method registerConsumer(eType, certificate-Path) on the LRC proxy in order to send the first parameter "eType" and he second parameter "certificate-path" that event consumer can prove its authorization suitable for the event type "eType" to the LRC. After proving that the certificate-path is valid, the LRC creates a session key and encodes it. Then it stores the event type, the public key of event consumer, just created session key, nonce, and lease-time in order to check consumers' authorization later on. As the values of results, this methods returns encoded session key and lease-time to consumer.



  However, event consumer, running low cost equipment, calls registerConsumer(eType, cert-List), to send all of the certificates which it owns.

- *registerListener(nTemplate, Listener, principal, sessionID)*

  After checking consumer's authorization and if the result is true, the LRC updates nonce and registers remote listener with the JES.



4.1.3 SSCM

By using certificate list sent by the event consumer of low cost equipment, the SSCM prove that the consumer can achieve the event instance of the event type. Also it checks the proof of certificate-Path directly sent by the event consumer in machines with powerful computing power. As a result, the SSCM includes the algorithm of "Certificate Chain Discovery"[10] and returns Boolean value of each case to the LRC(Listener Registration Controller). The SSCM interfaces are as Figure 3.5.

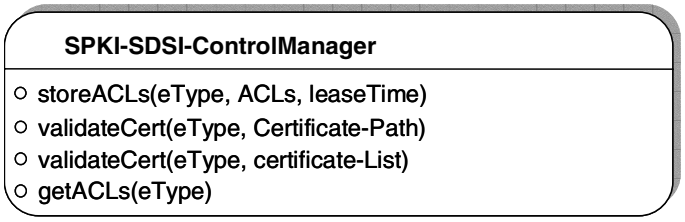


Fig. 3.5. Interfaces of the SSCM

- *storeACLs(eType, ACLs, leaseTime)*

  The ERC calls this method of the SSCM module in order to store ACL which is a collection of authorizations for the consumer to achieve events provided by event supplier.
- *validateCert(eType, Certificate-Path), validateCert(eType, Certificate-List)*

  These methods are invoked by the LRC in order to request authorization proof of event consumer. The first method owns Certificate-Path as parameter, a collection of certificates proven by consumers and the second one owns Certificate-List, a collection of all name and authorization certificates held by consumers.
- *getACLs(eType)*

  This method finds and returns ACL which is necessary for consumers to achieve designated types of event

4.1.4 ERC

The ERC (Event Registration Controller) is responsible for checking the proof of event types and ACL which are provided by event suppliers. It also checks the replication of events. Furthermore, the ERC has a responsibility of transferring events

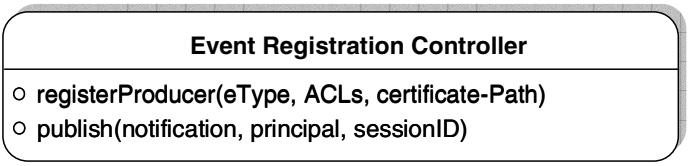


Fig. 3.6. ERC interface

provided by event suppliers to the JES. Interfaces provided by the ERC are shown in Figure 3. 6.

- *registerProducer(eType, ACLs, certificate-Path)*
  By sending an event type "eType", ACLs and authorization proof information "certificate-Path" to the SSCM, this method let the SSCM store them in its cash. Furthermore, this method creates encoded session key by using two parameters of producer's principal derived from certificate-Path and session key, a random nonce. This encoded session key is used to prove whether the event is authorized to provide itself to the JES or not.
- *publish(event, principal, sessionID)*
  This is the method that publishes suppliers' event. After checking if the event sent by suppliers is authorized and then if only if the return value is true, this method sends the event to the JES.

4.2 Testing of SeJES Service

This section summarizes how a consumer and a supplier use our SeJES service, implemented in Jini environment. In addition, it shows GUI screen that shows, in order to get event which the consumer is interested in, whether listener's registration in the SeJES is successfully committed or not. It also exhibits GUI screen including procedures and results necessary for suppliers to publish event.

4.2.1 Test Scenario

A scenario to test SeJES services is as follows:

Susan who possesses event supplier sensor1 sets ACL, <self, Bob, 1, "get event1", (05-11-20, 06-03-29)> in sensor1 in advance. When Susan turns sensor1 on, event supplier application in sensor1 registers event type(event1) and its ACL set by Susan with SeJES. In the meantime, Bob hopes to get the event1 published by gadget1 of Susan.

In order to get the event1 instance, Bob asks SeJES to send ACL corresponded with the event1. And then Bob calculates Certificate-Path establishing a chain from name and authorization certificates in its certificate cash to ACL of event1 which she wants to get. Now Bob registers his remote listener with SeJES and waits till the event arrives.

In the meantime, unauthorized Charlie also tries to register his remote listener to get the event1 of sensor1. In this case, since he can't prove his authorization, his request is rejected. Purchasing sensor2, Eve tries to use event service in order to publish event2, but it is rejected as well.

Figure 4.1 shows the order of methods invoked in order for event consumer to be notified and for supplier to provide the event.

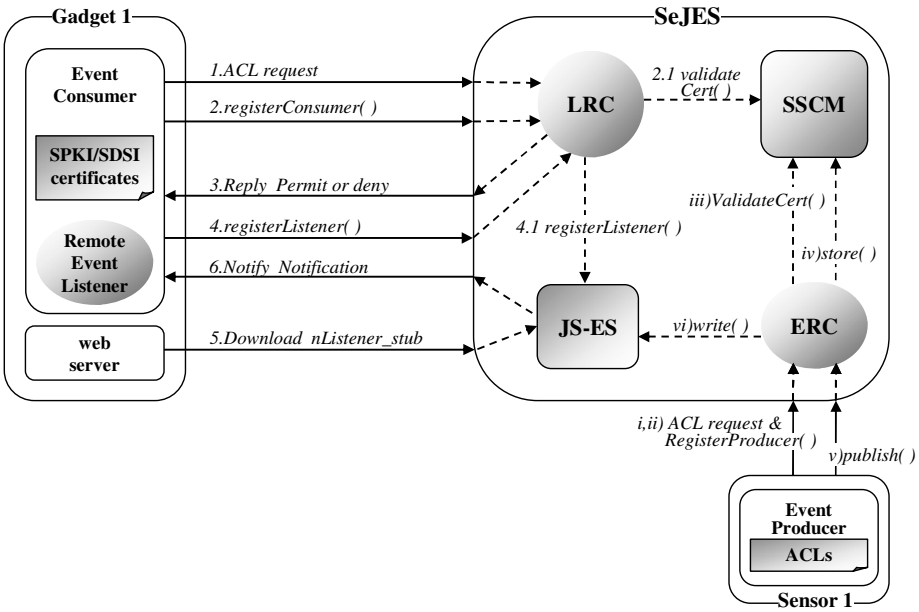


Fig. 4.1. Procedure of Secure Event Service

Figure 4.2 exhibits a screen dump showing that consumer Bob tried to register his listener with the SeJES in order to get an event "Alice location" and the SeJES notified those events to Bob.

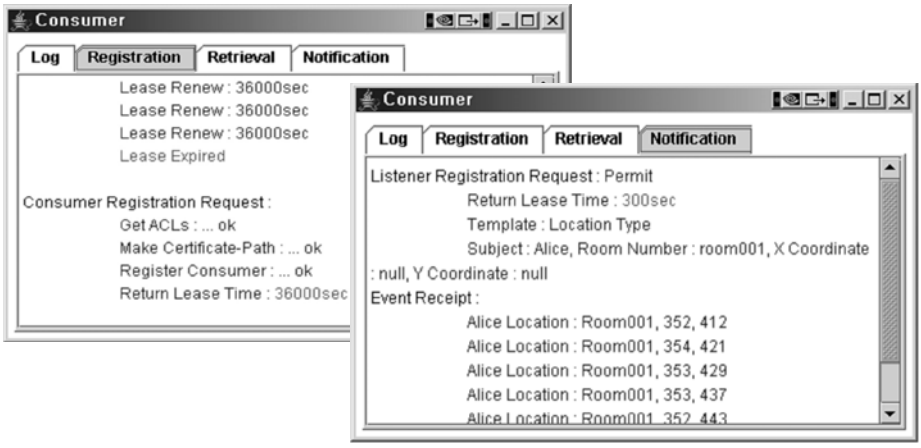


Fig. 4.2. Listener Registration & Event Notification

Figure 4.3 shows that the SSCM proves whether the certificate-path provided by the event consumer "Bob" is correct or not.

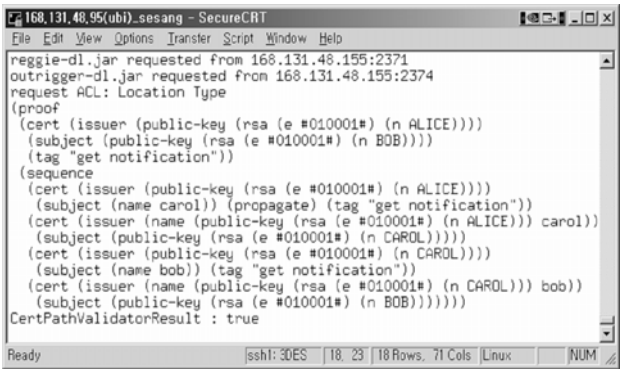


Fig. 4.3. Proving process of authorization certificate-path

Figure 4.4 shows the event producer of gadget1 Susan's log of event publishing.

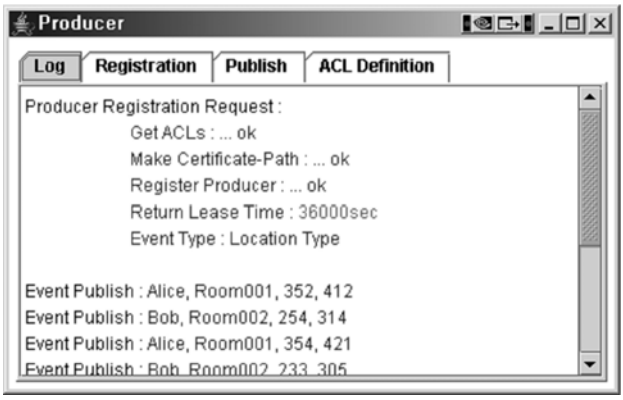


Fig. 4.4. Example the Screen of Event Publishing

5 Characteristics of Our SeJES System

SeJES is distinguished with CORBA in three aspects. Firstly, event service based on CORBA uses event channel. Accordingly, it is not totally an isolated model because event consumers and event suppliers communicate each other through the channel. Secondly, event service based on CORBA transfers its filtering responsibility to consumer programmers. Therefore, any events written in the channel are to be transferred to every consumer listening to the channel, which is an inefficient process by increasing excessive expenses of communication. However, our system can deliver only necessary events to a particular consumer in need because content based filtering using JavaSpace is available in the SeJES system. Finally, no persistency is existed in CORBA based event service. If the channel is down, it loses all information concerning consumers and suppliers connected to the channel. For that reason, there is no

way for consumers to take events provided while the channel is disconnected even after they are successfully connected again. However Our SeJES system is able to do that because of its *lease* () function.

6 Conclusions and Further Work

In this paper, we design and implement a secure event service, SeJES. Event consumers are able to register their listeners with the SeJES and disconnect their listeners at any time. Furthermore, the SeJES enables only authorized consumer and supplier to securely exchange their events by using SPKI/SDSI certificate. In addition, our SeJES guarantees that event is transferred to only all of the authorized event consumers and enhances the security by delivering event consumer and suppliers secure session key. The proposed event service, SeJES, is able to store event for lease-time. While being leased, event consumer commits a filtering based on content during the lease-time.

However, our SeJES is not able to federate event servers and does not provide the event service with priority. Also we want to implement our SeJES which has more QoS and more Filtering functions. Most of them are our future work.

References

1. Object Management Group: CORBAServices: Common Object Services Specification. Revised Edition. (1995)
2. Philip Bishop , Nigel Warren: JavaSpace IN PRACTICE. Addison-Wesley (2003)
3. Lee, Younglok., et al.: Development of Event Manager and its Application in Jini Environment. EUC Workshops 2005, LNCS 3823, Springer-Verlag, Nakasaki (2005) 704 – 713
4. Paul Stephens: Implementation of the CORBA Event Service in Java. A Thesis for the Degree of Masters of Computer Science, Trinity College, Dublin (1998)
5. IONA: OrbixEvents Programmer's Guide. IONA Technologies PLC (December 1997)
6. 6. IONA: OrbixTalk-The White Paper. Technical Report, IONA Technologies PLC, April (1996)
7. OUTBACK: jEVENTS-Java-based Event Service User's Guide. OutBack Resource Group Inc. (1997)
8. ICL Object Software Laboratories: DAIS Multicast Event Service. White Paper (1998)
9. Andrew J. Maywah: An Implementation of a Secure Web Client Using SPKI/SDSI Certificates. Master Thesis, M.I.T, EECS (2000)
10. Dwaine Clarke, Jean-Emile Elien, Carl Ellison, Matt Fredette, Alexander Morcos, Ronald L. Rivest: Certificate Chain Discovery in SPKI/SDSI. Journal of Computer Security, 9 (2000) 285-322
11. Sun Microsystems: .Jini™ Architecture Specification. Sun Microsystems, (1997-2000)
12. Keith Edwards, W., Edwards, W.: Core Jini. Pearson Education, (2000)