Directory Supported Management with SNMPv3

Salima Omari[1], Raouf Boutaba[2], Omar Cherkaoui[3]

[1] Laboratoire PRiSM, Université de Versailles, 45 avenue des Etats-Unies,
78 000 Versailles, France
osa@prism.uvsq.fr
[2]Computer Science Department, University of Waterloo,
Waterloo, Ontario, Canada, N2L 3G1
rboutaba@cs.uwaterloo.ca
[3] Laboratoire de téléinformatique, Université UQAM, CP 8888, Succursale Centre-Ville,
Montreal, Quebec, Canada, H3C3P8
cherkaoui.omar@ info.uqam.ca

Abstract. Data security and maintaining system integrity are the primary concerns pointed out by corporations and individuals when connecting to the Internet. To respond to this demand, the most recent agreed Internet management standard, SNMPv3, introduces new security features to make SNMP-based management ready for enterprise management. This is possible only if the SNMPv3 management framework is introduced properly in the enterprise, i.e., in such a way to respond efficiently to the management and security requirements specific to this enterprise. This paper proposes the use of standard directory service and protocol to configure SNMPv3 entities, to regulate SNMPv3 management exchanges and to customize security features according to the enterprise needs. It addresses in particular the configuration of access control parameters to be implemented by SNMPv3 entities according to enterprise security policies.

1 Introduction

Data security and maintaining system integrity are the primary concerns pointed out by corporations and individuals when connecting to the Internet. The recently agreed version of the Internet management standard, SNMPv3, overcomes the weakness of the previous versions of the protocol by allowing authentication, privacy and access control to be performed during the exchange of messages between SNMPv3 entities (i.e., SNMPv3 managers and agents). This functionality is performed within the entity subsystems, more precisely by the security subsystem and the access control subsystem. The modular architecture of the entity, as defined by the IETF (Internet Engineering Task Force), allows these subsystems to implement different security models. The models currently defined with SNMPv3 management are: USM (User-based Security Model) [6]; and VACM (View-based Access Control Model) [5]. These additions allow the SNMPv3 management platform to claim reliability and hence

R. Stadler and B. Stiller (Eds.): DSOM'99, LNCS 1700, pp. 121-132, 1999.

readiness to be introduced into enterprise network management. In current practices, configuration information and security parameters are maintained in distinct data stores, in multiple formats, and possibly by different managers. This may lead to inconsistent management decisions, which may lead, in turn, to the enforcement of conflicting security policies, and hence to jeopardizing enterprise information security.

This work aims at tackling the above-stated problem by using a uniform, enterprise-wide directory service to efficiently support SNMPv3 management applications. It aims in particular at implementing consistent models for security parameters within a logically centralized, physically distributed information store (the directory). In general, this approach will reduce the complexity of the task of managing large corporate networks. In particular, it will facilitate the proper configuration of SNMPv3 entities and ultimately its automation. Central to the proposed approach is the directory service used to store and access information about users, network devices, and applications including management applications. The adopted directory service allows various network applications and services to share information accessed through a uniform protocol, namely the Lightweight Directory Access Protocol (LDAP).

This paper emphasizes the use of a standard directory to store and access SNMPv3 security parameters used to configure and (re-)initialize SNMPv3 management entities. To promote interoperability between various management applications, SNMPv3 informational model is described according to the Directory Enabled Network (DEN) specification [3]. The DEN is an industry initiative standardized within the DMTF (Desktop Management Task Force) to provide a uniform model representing users, profiles, applications and network services within the directory through which management data from any source can be accessed in a common way. We extend the DEN specifications to integrate the SNMPv3 information model. SNMPv3 persistent information is maintained by the directory service and can be shared/reused by a variety of authorized management applications and tools.

This paper is organized as follows. Section 2, gives an overview of SNMP version 3 highlighting the security features, the subsystems of the SNMPv3 framework supporting these features, as well as the security models defined for these subsystems. Section 3, describes the usage of the directory for management purpose including the DEN specifications for enterprise network and systems management and the LDAP protocol commonly used as directory service and protocol. Section 4, introduces our proposal for SNMPv3 directory-enabled management. It presents an integrated SNMPv3 directory-enabled management architecture describes the LDAP classes related to the SNMPv3 entities. Section 5 describes our ongoing implementation and section 6 concludes the paper.

2 SNMPv3 Management Framework

Like the previous versions, the new generation of the Internet-Standard Management Framework (referred to as SNMP version 3) is based on the agent-manager model. However, it provides a new architectural model and new security features. The new features of SNMPv3 include security (authentication and privacy , as well as authorization and access control) and administrative framework (naming of entities, usernames and key management, notification destinations, proxy relationships, remotely configuration via SNMP operations).

The SNMPv3 specifications of the Internet-Standard Management Framework are based on a modular architecture [7]. As depicted by Figure 1, the SNMP entity, either manager or agent, consists of an SNMP engine and one or several associated applications.

The SNMPv3 engine consists of the dispatcher, the message processing subsystem, the security subsystem, and the access control subsystem.

The dispatcher coordinates the communications between the various subsystems. The message processing subsystem is responsible for preparing outgoing messages and for extracting data from received messages. It may support several message processing models, for example SNMPv1, SNMPv3, etc.

The security subsystem provides message security services such as integrity, authentication and privacy. Multiple security models may be supported by the security subsystem. For instance, the User-based Security Model (USM), described in RFC2574 [6], is the standard security model currently used with SNMPv3. It provides integrity, authentication and privacy services by computing message authentication codes, key management and data encryption. The access control subsystem constitutes a decision making point to allow or not a specific type of access (e.g., read, write, notify) to a particular object instance. Similarly, to the security services, the access control services can be provided by multiple access control models to allow future updates in case the security requirements change. The View-Based Access Control Model (VACM), described in RFC2275 [5], is one such model currently used within SNMPv3 access control subsystem. It basically allows restricting access to a subset of the management information referred to as a MIB view.

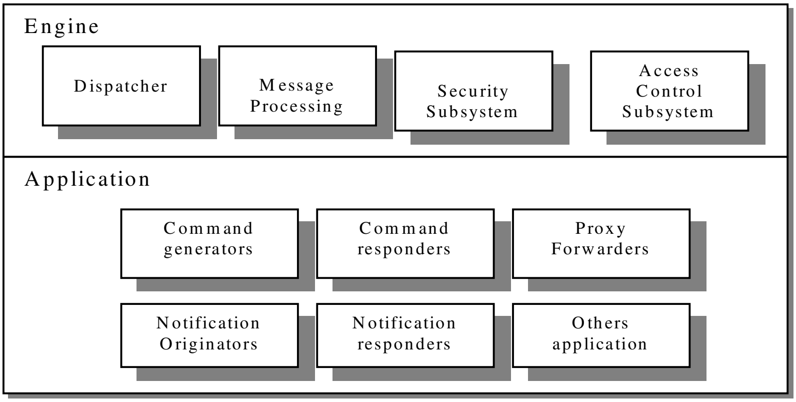


Fig. 1. Architecture of SNMPv3 entities

At the application level of the SNMPv3 entity, the various applications use the services provided by the SNMPv3 engine to accomplish specific tasks. According to the role of the SNMP entity (manager or agent), five dominant types of applications can be enumerated: command generators; command responders; notification generators; notification receivers; and proxy forwarders. The reader can refer to [8] for more details about these application types or other possible applications.

The subsystems, models, and applications within an SNMP entity may need to maintain their own sets of configuration information. Portions of the configuration information may be accessible as managed objects. The collection of these sets of information is referred to as an entity's Local Configuration Datastore (LCD).

As described in this section, the new SNMP framework offers an extensible architecture composed of a set of subsystems. Each subsystem may implement different mechanisms and support multiple models. Traditionally, security parameters are maintained in the MIBs encapsulated by the SNMP entities associated with the various network elements. This makes the maintenance of global security information difficult. Indeed, the various MIBs are distributed throughout the enterprise network and are more adapted for the storage of dynamic, frequently changing, information about the state of the devices. For persistent global management information such as enterprise security policies, a more appropriate repository is required. A directory service, that provides a logically centralized physically distributed database for enterprise wide information and global management strategies, is a more natural alternative. In this perspective, we introduce, in the next section, the use of a directory service for network management.

3 Directory Services for Network Management

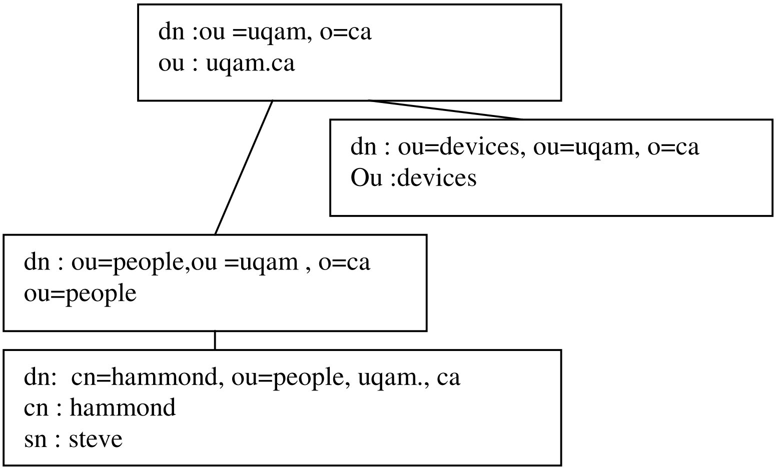
A Directory can be considered as a special database that contains attribute-structured information. Such a database is more descriptive than traditional relational databases. The information in a directory is generally read much more often than it is modified. As a consequence, directories don't usually implement the complicated transaction or roll-back schemes that regular databases use to do high-volume complex updates. Directories are tuned to give quick-response to high-volume lookup or search operations. Each object is represented in the directory by a set of attributes. For instance, an user object is represented in the directory by attributes such as name, address telephone number and so on. In addition, the directory service provides dynamic binding between addresses, application profiles, user names, and other information data stores. Historically, directory-type information was often stored in an application-specific private database, possibly shared across small groups through LAN file sharing using some kind of proprietary protocol. These barriers to global access are removed with the Lightweight Directory Access Protocol (LDAP), the Internet directory protocol defined by the Internet Engineering Task Force (IETF).

LDAP is a strategic directory protocol that runs over TCP/IP [10] [11]. It defines a reasonably simple mechanism for Internet clients to query and manage an arbitrary database of hierarchical attribute value pairs. The LDAP directory service model is based on entries. An entry is treated as an object. An LDAP object is an entity that is described with attributes and has a unique name, Distinguished Name (DN), by which it is referred to when used along with other objects. LDAP directory entries are organized in a hierarchical, tree-like, structure that reflects corporate, geographical and/or organizational boundaries. For instance, the entries representing countries appear at the top layer of the tree.

The definition of an LDAP object class includes, a name that uniquely identifies the class, an OID that also uniquely identifies the class, a set of mandatory attributes types, a set of allowed attribute types, and a Kind (structural, auxiliary, or abstract).

The kind of the object class indicates how the class is used. Structural classes, for example, describe the basic aspects of an object. It can be used to place restriction on where an entry can be stored within the directory information tree (DIT). Most object classes are structural classes. Auxiliary classes do not place restriction on where an entry may be stored, and they are used to add a set of related attributes to an entry that already belongs to a structural class. Abstract classes are rare and are used only for classes needed to support LDAP's basic information model.

Example of LDAP classes:


dn :ou =uqam, o=ca
ou : uqam.ca
dn : ou=devices, ou=uqam, o=ca
Ou :devices
dn : ou=people,ou =uqam , o=ca
ou=people
dn: cn=hammond, ou=people, uqam., ca
cn : hammond
sn : steve

The version 3 of the LDAP protocol [12] requires that directory servers publish their supported schemas through LDAP itself. This allows directory client applications to programmatically retrieve the schema and adapt their behavior based on it. It also allows the protocol to be extended to support new operations, and controls may be used to extend existing operations.

The authentication process, when accessing the directory, is called *binding*. There are different types of binding methods. In a simple bind, the client presents a DN and a password in clear text to the LDAP server. The server verifies that the password matches the password value stored in the *userpassword* attribute of the entry and, if so, returns a success code to the client. The simple bind does send the password over the network to the server in clear. However, one can protect against eavesdroppers intercepting passwords by encrypting the connections, for example using secure sockets layer (SSL) or transport security layer (TLS) [14]. SSL and TLS allow encrypting all the data flowing between a client and a server. LDAPv3 also includes a new type of binding operation, the SASL bind. SASL[13] is an extensible, protocol-independent framework for performing authentication and negotiation of security parameters. With SASL, the client specifies the type of authentication protocol it wants to use (e.g. kerberos, S/Key). If the server supports the authentication protocol, the client and server perform the agreed-upon authentication protocol. Incorporating SASL into LDAPv3 means that new authentication methods can be easily implemented for LDAP without requiring a revision of the protocol.

After the server has verified the identity of the client. It can choose to grant additional privileges based on some site-specific policy thanks to the access control list (ACL). ACL provides information about the entities to which those rights are granted, and the directory entries to which those rights apply. For example, a more permissive policy allows some authenticated users to modify certain attributes of their own entries whereas other users (e.g. administrative staff) may modify any attribute of any entry.

Actual implementations of directory services support a wide scope of applications including electronic mail, printing services, and many others. The LDAP directory service has also been used as a support for the management of network equipment. It allows both network administrators and routers to store and retrieve network-related information from a single point. This allows a network administrator to manage a network more efficiently compared to the case where persistent information such as configuration information is maintained in different databases at the levels of individual devices. Sharing management knowledge between various management processes is another advantage of using a directory service. Indeed, the directory acts as a global repository storing information about objects that are physically distributed.

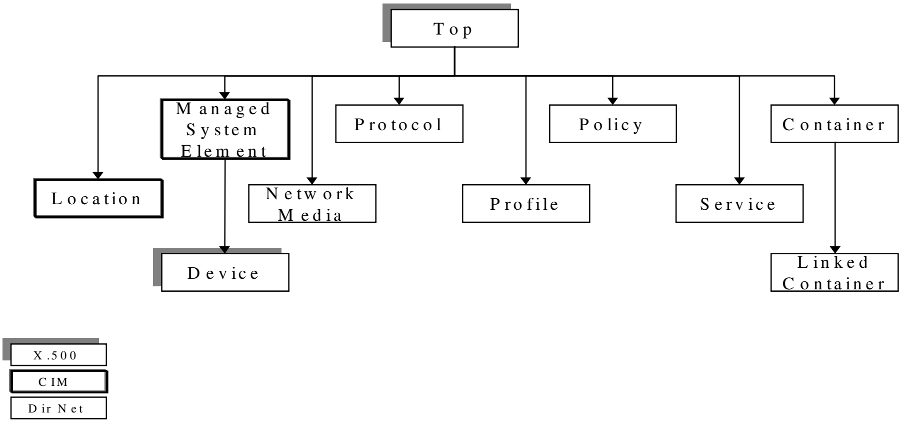


Fig 2. DEN Hierarchy Classes

Usually directory information has to be replicated in order to allow for efficient access from anywhere in the system as well as fault tolerance. It is the role of the directory service to ensure the consistency of the replicated information by reporting updates among the replicas. However, in order to permit a true exchange of network, system and service management data among heterogeneous management tools, a standard schema for describing these data is required. A recent initiative known as the DEN (Directory Enabled Network) standardized within the DMTF aims at enhancing the standard X.500 directory service to integrate enterprise network elements, services, and policies. The considered classes concern high-level management information such as routing policies or quality of service (QoS) parameters.

Fig.2. shows the main classes specified as part of the DEN initiative. It particularly shows how DEN is built upon CIM (Common Information Model) and X500 [9] to model the functional and management interfaces of network elements and services. The resulting common schema yields increased integration of management applications and tools.

4 Directory Support SNMPv3 Management

The objective pursued here through SNMPv3/DEN integration is the efficient configuration of SNMP entities. Rather than making a connection to each device and performing a one-by-one configuration of SNMPv3 entities, the network manager only places the appropriate configuration attributes into the node's entry in the directory server. When a SNMPv3 device is initialized, its associated LDAP client queries the directory server and obtains its configuration details. This allows configurations of the Local Configuration Data Store (LCD) with consistent information. The configuration is applied uniformly across different SNMPv3 entities.

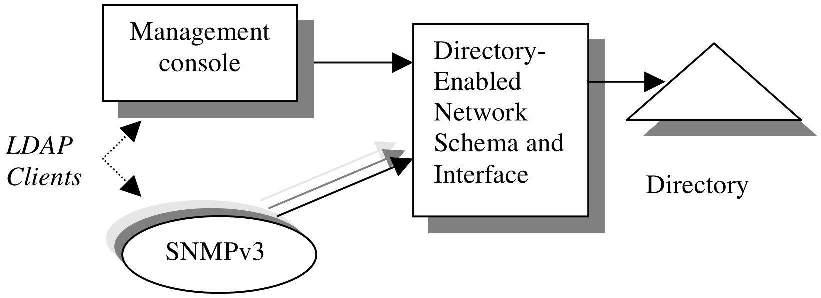


Fig. 3. LDAP-enabled SNMPv3 management

Another advantage of the directory-based SNMPv3 management is to enable various applications to share information on SNMPv3 entities through the directory. Fig. 3 illustrates the overall architecture including the directory service and SNMPv3-based management entities.

According to this architecture, the LDAP directory server contains the SNMPv3 LDAP classes. The Directory enabled network schema and interface consists of an API that allows clients to access the directory information.

There are two kinds of clients that access the directory:

- The administrator who populates the directory with data related to the snmpv3 entity. There are two types of data. The first type is related to the general configuration of the SNMPv3 entity subsystems, namely the security subsystem, the access control subsystem, the message processing subsystem, and the application part. These data are contained in the SNMPv3Modules class. The second type is related to “the existence” of the SNMPv3 agents and their individual configurations. A manager who wants to create an SNMPv3 agent must create a class with an engineID as distinguished name, and must decide its initial configuration. The references to the subsystems’ configuration of this SNMPv3 agent are then made automatically.
- The SNMPv3 entity. At entity set-up, the SNMPv3 application acting as an LDAP client searches its initial configuration into the LDAP directory server using the engineID. The SNMPv3 obtains the initial configurations and then translates them into SNMPv3 MIB attributes.

Dynamic configuration

To allow for dynamic and timely updates of SNMPv3-entities’ configuration, the configuration process interacts with a monitoring service, which in turn interacts with the LDAP directory service. The provided event-based monitoring service allows to associate data from a variety of applications, and allows applications such as configuration management to capture the flow of information and respond to events as they occur. Directory principals, i.e. users, applications, and network devices, store event types in the directory and produce events that are stored in event queues. After setting up the initial configuration, the application registers itself to receive notifications from the server whenever the data into the directory classes related to SNMPv3 configuration are updated. In the latter case the configuration application query the directory server to retrieve the details of the occurred event and the configuration changes. Based on these information, the configuration management application can take the appropriate re-configuration actions on the network devices.

Security services

Security procedures are performed thanks to the security and the access control subsystems in the SNMPv3 entity. USM, implemented by the security subsystem, and VACM, implemented by the access control subsystem, are the models currently standardized as part of the SNMPv3 framework. The USM model provides authentication, which checks the identity of data sources. It also provides privacy through a protection against disclosure of the message payload, and a protection against message delay or replay. Management operations using the USM model use the traditional concept of a per-user association of security information. An SNMP engine must have knowledge of any particular user from which authorized management operations are stemming prior to perform these operations. The VACM model allows checking if access for a given group is permitted to a set of MIB variables identified by a MIB View.

The use of a directory service in supporting SNMPv3 management should not threaten SNMPv3 security. The LDAP directory protocol must provide a sufficient security level so as not to jeopardize the SNMPv3 security feature. The LDAP security model relies on the connection-oriented nature of the LDAP protocol. This means that a LDAP client opens a connection with an LDAP server and sends a number of protocol data units on the same connection. The clients are responsible for initiating a communication to an LDAP server using the bind operation, which involves an authentication of the client. A client identifies itself by providing a distinguished name (DN) and a set of credentials. The server checks whether the credentials are correct for the given DN and, if they are, notes that the client is authenticated as long as the connection remains open.

4.1. SNMPv3-LDAP Objects

The integrated SNMPv3/DEN schema extends the DEN information model by adding classes that represent SNMPv3 entities. This integration facilitates the management of network devices offering SNMPv3 interfaces. The LDAP directory contains configuration parameters of SNMPv3 entities. These SNMPv3 data are searched and accessed (read/write operations) by management entities directly or by network administrators through a management tool capable of parsing and displaying the specific schema structure.

We have defined two kinds of LDAP object classes related to SNMPv3 management. The first one describes the SNMPv3 entities that must exist in the administrative Network. This object class, see fig.4, identified by the SnmpEngineID. The latter uniquely identifies the SNMPv3 entity and hence the representative object of the class. The attributes of the object class contain the description of the entity and its subsystems.

The second kind of object classes relates to the different model and subsystem that can compose the entity and their possible initial configurations. The manager has to select an initial configuration. These classes are ACM, SM, MP, D and Application classes which respectively correspond to the access control model subsystem, the security model subsystem, the message processing subsystem, the dispatcher subsystem, and the applications. Since each subsystem may contain several models, we define sub-classes, each of which corresponds to a given model. Examples of these sub-classes are the VACM and USM sub-classes. The MP class contains others sub-classes, each of which describes the message processing model, such as the one of SNMv1, SNMPv2, or SNMPv3. Fig.4 gives an overview of the LDAP object classes related to the SNMPv3 entity.

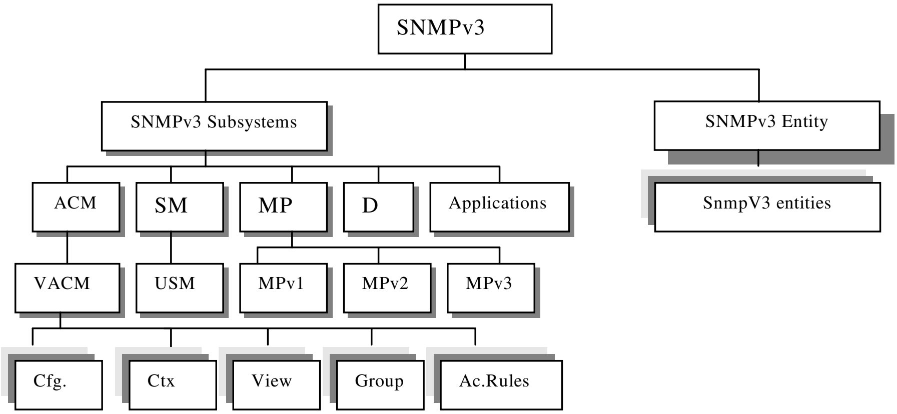


Fig.4. SNMPv3 LDAP object

The modularity of the SNMPv3 architecture and the object oriented modeling of information allow to add (respectively remove), in a flexible way, applications into (respectively from) an SNMPv3 entity either manager or agent. Our directory based management architecture allows to easily incorporating a directory service client (LDAP client) into the entity application part (see Fig.1). This new application is launched when the SNMPv3 entity is started within a device. It requests the directory service (LDAP server) for entity information such as entity configuration parameters or security parameters. Fig.5 shows the extension of the SNMPv3 entity architecture to include an LDAP client and thereby integrating directory service support for SNMPv3-based management applications.

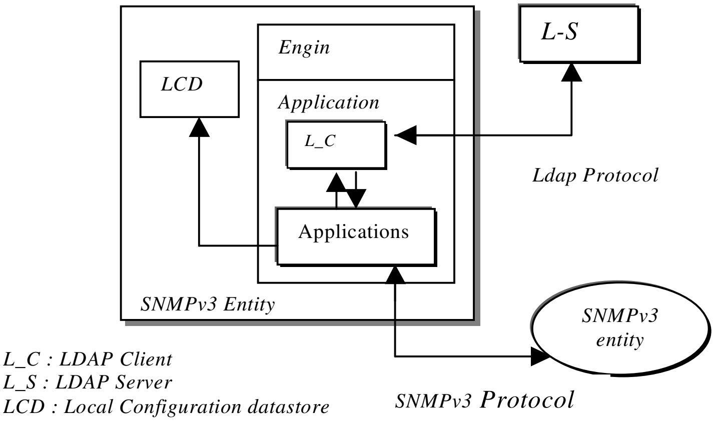


Fig. 5. SNMPv3 entity architecture

The implementation of the prototype is done at UQAM University as an extension of the ModularSNMPv3 project, which implemented a SNMPv3 platform in JAVA.

To configure the SNMPv3 entities we have used the Sun Directory which offers a global directory and naming service. The latter includes an LDAP server compliant with the LDAP v3 Internet standard. It provides options to encrypt the communications between clients and the server using SSL, and an authentication mechanism (CRAM MD5) during the binding operation through the SASL protocol. Also supported is the EXTERNAL mechanism when the SSL library is installed on the server, and the server is configured to support TLS security.

The Java Naming and Directory Interface (JNDI) [15] has been used for the implementation of the LDAP clients, the user interface and the SNMPv3-configuration application. JNDI provides directory and naming functionality to Java applications. It is defined to be independent of any specific directory service implementation. Consequently, a variety of directories already existing or upcoming can be integrated and accessed in a uniform way.

Finally, the EventDirContext method contained in the Javax.naming.event package provided by JNDI2.1 is used to permit the registration of clients' interest in the occurrence of certain events and their notification whenever these events occur. This way LDAP clients representing SNMPv3 entities are dynamically notified when configuration information is updated in the directory. The updates are then retrieved from the directory and the corresponding changes are performed as part of the entity reconfiguration.

5 Conclusion

This paper has shown the benefit of using a directory service to support SNMPv3 management applications and to promote interoperability between the various enterprise applications. It has shown how the directory can be populated with enterprise wide information that is relevant to secure SNMPv3 management. In particular, the directory has been used to store and access persistent SNMPv3 entities' configuration information. This approach allows customizing the access control and security models (VACM and USM) implemented by the entities according to the enterprise-specific security requirements. Interoperability is achieved on one hand by using the widely accepted LDAP standard protocol for interacting with the directory service and on the other hand by adopting a common schema for describing directory information. We have specified the SNMPv3 management entities using the LDAP classes. To integrate directory service support for our SNMPv3 management system, we have implemented LDAP clients within the application layer of our SNMPv3 entities. The configuration of an SNMPv3 engine, one or several of its subsystems, is conducted through the associated LDAP client application by accessing and retrieving configuration information from a logically centralized LDAP server.

The major addition to SNMP-based management that is brought by the new version of the protocol (SNMPv3) is the security feature. Therefore, we need to emphasize the security issue by using the directory service to store and access enterprise wide security policies and to configure access control and security subsystems of the

SNMPv3 management entities. In [4], we have defined a model for expressing and storing management policies as well as its mapping into SNMPv3 access control and security models, particularly VACM and USM. As a future work, we intend to use the directory service to store and maintain enterprise wide management policies. This will allow for enterprise policy-driven management and customized and flexible control of SNMPv3 management application. For example, security policies will be flexibly added, removed and modified by accessing the directory service, instead of being hard-coded into security management applications. The directory supported configuration service of SNMPv3 entities described in this paper will be extended to ensure the mapping of high-level enterprise security policies into SNMPv3 security parameters and to detect/resolve/prevent policy conflicts.

References

1. Case, J., M. Fedor, M. Schoffstall, and J. Davin, "The Simple Network Management Protcol", STD 15, RFC 1157, University of Tennessee at Knoxville, Performance Systems International, Performance International, and the MIT Laboratory for Computer Science, May 1990.
2. Jeffrey D Russ Mundy David Partain Bob Stewart, «Introduction to Version 3 of the Internet-standard Network Management Framework» Draft, Octobre 1998.
3. Judd, S., Strassner, J., "Directory Enabled Networks - Information Model and Base Schema", Draft version 3.0c5, August 1998.
4. Omari, S., Boutaba, R., Cherkaoui, O, "Policies for SNMPv3-based management", IEEE /IFIP IM'99, Mai 1999.
5. Wijnen, B., Presuhn, R., and K. McCloghrie, "View-based Access Control Model for version 3 the Simple Network Management Protocol (SNMP)", RFC 2275, January 1998.
6. Blumenthal,U., Wijnen, B., "User-Based Security Model for version 3 of the Simple Network Management Protocol (SNMP)", RFC 2574, April 1999.
7. Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for describing SNMP Management Frameworks", RFC 2271, January 1998.
8. Levi, D., Meyer, P., and B. Stewart, "SNMPv3 Applications", RFC 2273, January 1998.
9. ITU-T Rec. X.500, "The Directory: Overview of Concepts, Models and Service", 1993.
10. Wahl, M.,. Coulbeck, A., Howles,T., Kille, S., "Lightweight Directory Access Protocol Attribute SyntaxDefinitions", RFC 2252, December 1997.
11. Howes, T, Smith, M, Good, G, "Understanding and deploying LDAP Directory services", MTP edition
12 Wahl, M., Howes, T, Kill, S., "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1998
13. Myers, J., "Simple Authentication and Security Layer (SASL)", RFC 2222, October 1997
14. Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999
15. Java Naming and Directory Interface (JNDI)., www.java.sun/jndi