Cookbook

20th October 2003

(DRAFT version 0.45)

Abstract

The ViDeNet Video Middleware Cookbook provides its audience with a good understanding of the theory and use of middleware for videoconferencing and voice over IP (VoIP). The focus is understanding the new ITU-T standard, "H.350 : Directory Services Architecture for Multimedia Conferencing". The new H.350 standard describes a directory services architecture for multimedia conferencing using LDAP. Standardized directory services can support association of persons with endpoints, searchable white pages, and clickable dialing. The cookbook explains how a standardized LDAP schema called "*commObject*" (communications Object class) can be used to represent endpoints on the network and to associate those endpoints with users. Design and implementation considerations for the inter-relation of video and voice-specific directories, enterprise directories, call servers and endpoints are also discussed.

1 Acknowledgements

Video Middleware Cookbook Editors:

- Jill Gemmill, University of Alabama at Birmingham, Principle Investigator
- Jason L. W. Lynn, University of Alabama at Birmingham, Testbed Manager

Video Middleware Cookbook Authors (in alphabetical order):

- Nadim El-Khoury, University of North Carolina at Chapel Hill

- Jill Gemmill, University of Alabama at Birmingham
- Alisa F. Haggard, University of North Carolina at Chapel Hill
- Tyler Miller Johnson, University of North Carolina at Chapel Hill
- Jason L. W. Lynn, University of Alabama at Birmingham, Testbed Manager
- John–Paul Robinson, University of Alabama at Birmingham

Video Middleware Cookbook Contributors (in alphabetical order):

- Samir Chatterjee, Claremont Graduate University, Co-Principle Investigator
- Aditya Srinivasan, University of Alabama at Birmingham
- Henny Bekker, SURFNet, Directory Services
- Egon Verharen, SURFNet, Director of Innovation
- RADVISION

This material is based upon work supported by the National Science Foundation under Grant No. 0222710 to the University of Alabama at Birmingham. Additional support was provided by EPS-0096193, NSF ANI-0123937 via SURA-2002-103 Subcontract. H.350 was introduced in NMI Release 2 of the NSF Middleware Initiative as *commObject*, an NMI-EDIT component, in part supported by ANI-0123937. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Additional support has been provided by the University Corporation for Advanced Internet Development (Internet2). We acknowledge and thank the support of all our colleagues at Internet2 VidMid working group. Special thanks to Tom Barton, Michael Gettes, Bob Morgan, Sasha Ruditsky, and Art Vandenberg for their contributions to *commObject*, and to the Department of Physics at UAB for providing some of the equipment used in the Testbed.

2 Introduction

2.1 Overview

The ViDeNet Video Middleware Cookbook provides its audience with a good understanding of the theory and use of middleware for videoconferencing and voice over IP (VoIP). The focus is understanding the new ITU-T standard, "H.350 : Directory Services Architecture for Multimedia Conferencing" that describes a directory services architecture for multimedia conferencing using LDAP. Standardized directory services can support association of persons with

endpoints, searchable white pages, and clickable dialing. This cookbook explains how a standardized LDAP schema called "*commObject*" (communications Object class) can be used to represent endpoints on the network and to associate those endpoints with users. Design and implementation considerations for the inter-relation of video and voice-specific directories, enterprise directories, call servers and endpoints are also discussed.

Directory services can provide searchable white pages, automated configuration of endpoints, association of persons with endpoints, and user authentication based on authoritative data sources. These functions can result in reliable accounting and centralized endpoint management while improving the users' ability to locate and correctly dial other multimedia users. This cookbook also describes a way to represent endpoints on the network and to associate those endpoints with users and discusses design and implementation considerations for interrelating video-and-voice-specific directories, enterprise directories, call servers and endpoints.

A generic super class, called *commObject*, is used to represent attributes common to any video or voice protocol. The cookbook describes the formal LDAP object class definitions and configuration files for *commObject* and its subclasses *h323Identity*, *sipIdentity*, *h320Identity*, and *genericIdentity*. These classes can be used to represent h.323 endpoints, SIP user agents, h.320 endpoints, and users reachable via non-standard videoconferencing systems such as Access Grid and Virtual Rooms Videoconferencing System, respectively, in a directory.

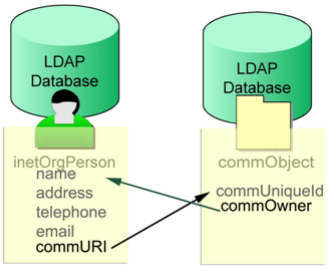
The *commObject* class is an abstraction of a video or voice over IP device. The *commObject* class permits an endpoint (H.323 endpoint or SIP user agent or other protocol endpoint) and all their aliases to be represented by a single entry in a directory. Auxiliary classes can be derived from *commObject* to represent specific protocols, such as h.323, h.235, or h.320. These subclasses are fully described in the ITU-T H.350.X series of Recommendations. Multiple H.350.X classes can be combined to represent endpoints that support more than one protocol. For example, endpoints that support H.323, H.235 and H.320 would include H.350, H.350.1, H.350.2, and H.350.3 in their LDAP representations.

There are two basic components in the architecture and they are used to relate a person directory entry to protocol specific information about their video or voice over IP device and to also relate a device to its owner. The commURI object is a class whose only purpose is to link a person or resource to a *commObject*. By placing a commURI 'pointer' in a person's directory entry, that person becomes associated with the particular targeted *commObject*.

Similarly, *commObject* contains a pointer, called *commOwner*, that points to the person or resource that owns or is associated with the *commObject*. In this way, people or resources can be associated with endpoints and endpoints can be associated with the people who use them.

Many organizations and enterprises already have an authoritative directory listing people associated with the enterprise. To support H.350, the only change required in the enterprise directory is the addition of the simple object class *commURI* (a labeled URI). *commObject* data may be instantiated in the same or in

Figure 1: 2.1



entirely separate directories, thus allowing implementers flexibility in deploying the architecture.

The Cookbook is organized as follows: PART 1 provides a summary of current standards for videoconferencing and voice over IP protocols and reviews common architectural deployments for systems adhering to those standard protocols. The H.350 Directory Services Architecture for Multimedia Conferencing is introduced, accompanied by a summary of the LDAP directory standard and review of authentication/security standards that are part of the H.323 and SIP protocols. PART 2 of the Cookbook takes the reader through a discussion of decision points that are part of deploying directory-enabled videoconferencing architectures and provides step-by-step installation and configuration instructions for Sun iPlanet, OpenLDAP, and Microsoft Active Directory directory servers. Each object class and attribute of the H.350 schema is explained, with examples. PART 3 is intended primarily for software developers. Code samples in several programming languages are provided as examples of how to use the H.350 directory service in endpoints and callservers for configuration, lookup, authentication and user authorization decisions.

2.2 Intended Audience

The intended audience for this cookbook consists of videoconferencing / voice over IP software developers and system administrators. This manual will be especially useful to developers of videoconferencing endpoints, user agents, proxies, gatekeepers, and gateways that are being designed or redesigned to function with middleware. In addition, this cookbook is intended for administrators of various kinds that perceive the need to implement a videoconferencing middleware infrastructure. This manual will provide an overview for managers and specifics for LDAP and videoconferencing administrators so that they may easily and painlessly implement videoconferencing middleware. With that said, the stated intended audience is not meant to limit the reading population and we welcome readers that have a general interest in this architecture.

The examples and configurations in this cookbook are derived from and intended for higher education institutions, though one may easily translate the

examples and configurations for other institutions by leaving out any steps that involve information specific to higher education.

This manual extends the ViDeNet architecture. ViDeNet is a testbed for the exploration of issues associated with globally scalable video and voice over IP deployments. ViDeNet provides experience leading to best practices and standardized operations achieved through consulation, collaboration and consensus.

3 Why *commObject*?

3.1 Problem Statement

The use of a common, authoritative data source for call server, endpoint, user, authentication and white pages information is important in large scale multimedia conferencing environments. By standardizing the LDAP schema used to represent the underlying data, products from different system vendors can be deployed together to create an overall application environment. For example, a white pages search engine developed by one provider could serve directory information to IP telephones produced by a second provider, with signaling managed by a call server produced by yet a third provider. Each of these disparate systems can access the same underlying data source, reducing or eliminating the need to coordinate separate management of each system. Management of this data can be incorporated into existing customer management tools, providing quick and flexible scaling up of applications.

Many technology providers have already incorporated LDAP into their products, but have had to do so without benefit of a standardized schema. H.350 is the first effort to standardize those representations. While URLs are already standardized for several conferencing protocols, their representation in a directory is not. H.350 supports a standardized way for URLs to be searched and located. This is a necessary step to support clickable dialing.

Management of endpoint configurations can be improved if the correct settings are stored by the service provider in a location that is accessible to both service provider and endpoint. LDAP provides a storage location that can be accessed by both call server and endpoint; thus it is possible to use the directory for automated endpoint configuration, which is important for simplifying operation and supporting user mobility.

3.2 *commObject* Design Goals

Large-scale deployments of IP video and voice services have demonstrated the need for complementary directory services middleware. Service administrators need call servers that are aware of enterprise directories to avoid duplication of account management processes. Users need white pages to locate other users with whom they wish to communicate. All of these processes should pull their information from canonical data sources in order to reduce redundant administrative processes and ensure information accuracy. The following design criteria

were used to establish this architecture:

1. Enable endpoint information to be associated with people. Alternately, enable endpoint information to be associated with resources such as conference rooms or classrooms.

2. Enable online searchable "white pages" where dialing information (e.g. endpoint addresses) can be found, along with other "traditional" directory information about a user, such as name, address, telephone, email, etc.

3. Enable all endpoint information to be stored in a canonical data source (the Directory), rather than local to the call server, to make use of authoritative data sources and avoid unnecessary replication.

4. Support the creation of very large-scale distributed directories. These include white pages "portals" that allow searching for users across multiple institutional directories. In this application, each enterprise directory registers itself with (or is discovered by) a directory of directories that is capable of searching across multiple LDAP directories.

5. Support multiple instances of endpoints per user or resource.

6. Represent endpoints that support more than one protocol, for example endpoints that are both H.320 and H.323.

7. Store enough information about endpoint configuration so that correct configuration settings can be documented to end users on a per-endpoint basis, as a support tool, or loaded automatically into the endpoint.

8. Be extendable to allow implementation specific attributes to be included.

9. Be non-invasive to the enterprise directory, so that support for multimedia conferencing can be added in a modular fashion without significant changes to the enterprise directory.

The scope of H.350 does not include extensions of functionality to existing protocol definitions. H.350 merely represents existing protocol attributes. Exceptions to this case occur only when functionality is implied by the directory itself, such as the commPrivate attribute that flags a "do not publicly list" directory entry.

3.3 H.350 Origins

H.350 has its origins in the academic research networking community. The Video Development Initiative (ViDe) has been working since 1998 to peer together video and voice over IP networks operated autonomously by various universities. This peering created a test bed called ViDe.Net in order to explore issues associated with the global deployment of video and voice over IP. To date over 100 universities, national research networks, backbone networks, and corporate networks peer together using H.323 in the test bed, creating an 'Internet of video and voice over IP.'

Early on in the development of ViDe.Net it became apparent that a white pages directory was a fundamentally important feature. The directory was needed not only to lookup the name and dialing address of other users, but also as an aid to configuration so that users could find their own address information and system administrators could contact users. Proprietary address books offered by participating vendors were inadequate because they did not include all the information required, were not portable across vendors, and were not searchable or manageable outside of their internal environment. Clearly there was a need for a standardized directory format.

As deployments grew larger, network managers grew services out of the test bed environment and into production. Since commercially available call servers typically featured internal databases of endpoints, the network manager found he or she had to separately manage, generally by manual data entry, each user of the service. This situation was observed to be one significant element keeping deployment sizes small because the cost of deployment involved a significant amount of labor to locate, verify, and store information about users and their endpoints. Enterprises, such as higher education institutions, already had directories of users containing information from authoritative institutional sources and that information was already managed by appropriate entities within the enterprise. It became evident that in order to scale voice and video over IP up to greater orders of magnitude, the multimedia conferencing environment should leverage the existing directory infrastructure.

Internet2 and ViDe together recognized the need to standardize a directory services architecture for video and voice over IP and jointly created the Video Middleware Working Group (VidMid-VC) to accomplish that task. Working together with experts in video and voice over IP protocols, as well as experts in middleware, security and directory technologies, VidMid created the *commObject* architecture and submitted the architecture to ITU-T Study Group 16. Study Group 16 received the architecture and decided to take it on as project H.LDAP. Because OID assignments were critical to creating the document, the document number H.350 was tentatively assigned to the project. H.350 was ratified by the ITU-T as an international standard in August 2003.

3.4 The ViDeNet Middleware Test Bed

The *commObject* architecture has been adopted in the ViDe.Net test bed in order to demonstrate that the concept is viable and to identify problems in the architecture. You are invited to use the test bed as an aid to understanding the H.350 architecture. The URL for the test bed is https://videnet.unc.edu/., where you can create both user accounts (people directory entries) and zone administrator accounts (for managing entire networks). There is no charge for using the test bed.

3.4.1 White Pages

The ViDe.Net implementation of *commObject* is a self-contained demonstration of the H.350 Directory services for Multimedia architecture. Figure 3.1 and Figure 3.2 show the linked relationship between the people directory and the *commObject* directory.

Figure 2: 3.1

Figure 3.1 shows a user entry in the people directory; the object class used for the person entry is the IETF inetOrgPerson object class. A commURI exists for each of the user's endpoints. The commURIs are listed in the 'Endpoint' section in the figure. Note that the label portion of the commURI is used to create a friendly name for each endpoint, such as 'Home Office'. The words used in the label can be assigned and changed by the user to provide a description of each endpoint. Clicking on the commURI labeled 'Laptop' brings up the actual *commObject* information for that endpoint as show in Figure 3.2.

Figure 3: 3.2

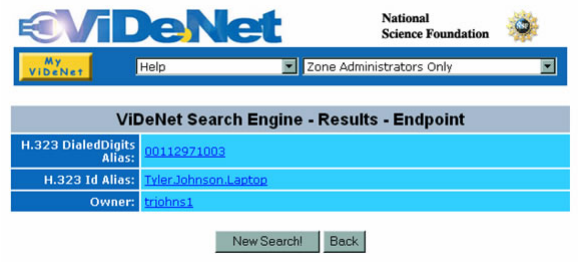


Figure 3.2 shows the *commObject* information for the selected endpoint. Note that the 'Owner' is a link that refers back to the user associated with this endpoint which is an H.323 videoconferencing device. H.323 information is stored using *the h323Identity* object class as described in H.350.1. Note also that these aliases are displayed in H.323 URL format so that clicking on the address could cause a call to be established for any endpoint supporting that functionality.

It is not necessary to have the people and *commObject* information displayed on separate panels in a user interface. This was an implementation choice made to graphically demonstrate the relationship between the two and to easily accommodate large numbers of endpoints associated with individual users.

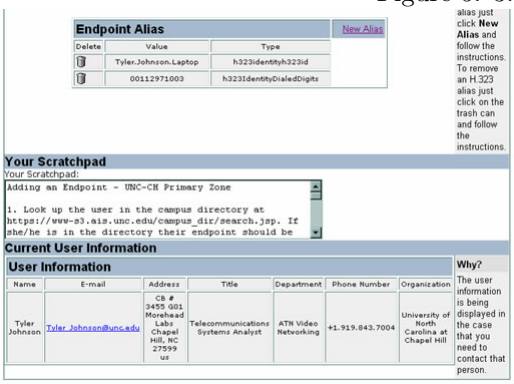
3.4.2 Network Management

The test bed also demonstrates the use of *commObject* to manage a network of multimedia conferencing devices. Each network administrator has secure access to the *commObject* data for their network. The administrator can manipulate this data and the subsequent changes are reflected in the directory. The administrator never has to manage user information. In the test bed, user information is managed directly by the user and, in an enterprise environment, user information is managed by a separate and likely pre-existing user management system.

Figure 4: 3.3

Figure 3.3 shows the choices available to a zone administrator. These choices take advantage of information already present in the enterprise directory. In particular, note that selecting 'Update, Remove an Associated Endpoint' will bring up the editing screen shown in Figure 3.4.

Figure 5: 3.4



In this example, the *commObject* information is displayed for editing by the administrator, while the user information is displayed for information purposes and is pulled from the enterprise directory.

4 Background Information

4.1 Videoconferencing and VoIP

4.1.1 Overview

Videoconferencing in its most basic form is the transmission of synchronized image (video) and speech (audio) back and forth between two or more physically separate locations, simulating an exchange as if the participants were in the same physical conversation. This is accomplished through the use of cameras (to capture and send video from your local endpoint), video displays (to display video received from remote endpoints), microphones (to capture and send audio from your local endpoint), and speakers (to play audio received from remote endpoints). Multi-participant conferences require either (a) a "multi-point control unit", which is a server receiving video and audio from each participant and transmitting image and audio of the active speaker back to each participant of (b) an IP multicast enabled network and application, where each participant sends voice and video to every other participant. An excellent introduction to videoconferencing is available in the ViDe Videoconferencing Cookbook.

Protocol standards for videoconferencing enable interoperability among different vendors' implementations of these components. While a number of videoconferencing solutions exist, this cookbook will focus on those that are standards based. Today, there are two standards for videoconferencing and voice over IP: the International Telecommunication Union (ITU) developed the earliest such standard, known as H.323, and the Internet Engineering Task Force (IETF) has developed the Session Initiation Protocol (SIP).

4.1.2 Standards for videoconferencing/VoIP

4.1.2.1 ITU-T H.323 The ITU Telecommunication Standardization Sector (ITU-T) is one of three Sectors of the International Telecommunication Union (ITU) and works to provide high quality standardsÂ (Recommendations) covering all fields of telecommunications. Arguably the most popular and extensible early compressed videoconferencing was enabled via the ITU standard called H.320, describing videoconferencing services over ISDN and T1 leased and dedicated telephone lines.

H.323, was first approved by the ITU in 1996. H.323 was designed to both function like and interoperate with H.320, changing the transport layer only so that the protocol would work on the Internet. H.323 parallels the telephone system in having an architecture based on intelligent call control and relatively "dumb" endpoints; this design is based on circuit-switching and a desire to provide centrally controlled, highly managed services.

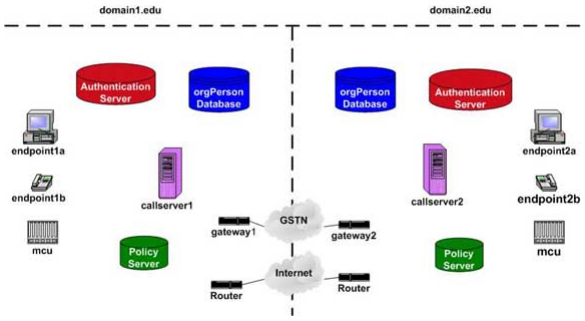
H.323 is not one particular protocol but is rather an umbrella standard consisting of several different protocols. Signaling and call control are handled by Q.931 as defined in Recommendation H.225.0 and Recommendation H.245. Once call control completes, the media transfer can begin by using the IETF Real Time Protocol (RTP). A variety of audio and video codecs are supported, and the standard includes both necessary and optional components. H.323 includes a security protocol known as H.235, which describes authentication and encryption using passwords or digital certificates.

4.1.2.2 IETF SIP SIP is a signaling protocol for establishing calls and conferences over IP networks. Unlike H.323, SIP is a standalone protocol; it does not handle media transfer, resource reservation, or even session description. Even though SIP is not an all-in-one solution, it does work well with other protocols and thus, allows for flexibility in defining a videoconferencing solution. RTP may be used for media transfer and SDP for session description. SIP originated in the mid 90's (about the time H.323 was becoming finalized as a standard) so that it would be easy to invite people to view an IP multicast session like a shuttle launch on the M-Bone.

Rather than being a telephony-based protocol, SIP is modeled after other Internet text-based protocols such as SMTP and HTTP, and was designed to establish, change, and tear down calls between one or more users in an IP network in a manner totally independent of the media content of the call. Like HTTP or eMail, SIP moves application control to the endpoint, eliminating the need for intelligence in the network core. This design is based on TCP/IP and a desire to put as much management control as possible at the endpoint.

The SIP protocol includes authentication using passwords or signed (S/MIME) messages. The SIP protocol is fully described by IETF RFC 3261.

Figure 6: 4.1



4.1.3 Videoconferencing and VoIP Architecture Overview

The basic architectural components for videoconferencing/Voice over IP are illustrated in Figure 4.1. This diagram is protocol-independent and emphasizes the similarities between H.323 and SIP architectures.

Two independently administered and geographically separated domains are depicted. Each domain contains endpoints, an authentication server, a person directory, a call server, a policy server, one or more gateways and routers. Some components, such as endpoints and callservers, are commonly deployed; other components such as policy servers are rarely deployed at this point in time.

For both H.323 and SIP architectures, users have a device such as a mobile phone, PC or laptop with microphone and camera, or other "appliance". All these devices contain the client software for the protocol. The H.323 protocol refers to these clients as "Endpoints"; the SIP protocol refers to these clients as "User Agents". These clients are able to directly place calls to other clients of the same protocol. The client software also encodes and decodes the multimedia streams.

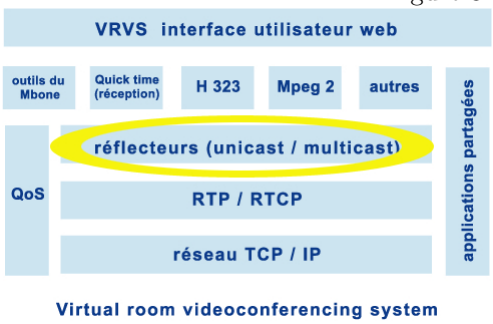
Both H.323 and SIP architectures utilize a call server. The H.323 protocol refers to this server as the "Gatekeeper" while the SIP protocol refers to "Proxy Servers". Call servers are used for signaling information outside of the media stream, including user authentication, managing usage, user's location, etc.

The Multipoint Control Unit (MCU) is used for multiparty conferencing. In H.323, the MCU is treated as a special endpoint; it's role is to receive incoming streams from each conference participant and select one of these streams to send back to each conference participant except the one selected for this broadcast. SIP MCU's are in early beta testing at the time this cookbook was written.

Gateways serve as protocol translators; to place a call from an H.323 endpoint to a SIP user agent, an intervening gateway would be needed to translate both the call setup/session initiation steps; both H.323 and SIP use RTP to encode the media.

Figure 7: 4.2

Figure 8: 4.3



4.1.4 Non-standard Videoconferencing Protocols

Additional, non-standard protocols for videoconferencing also exist. Two popular systems used especially in higher education research circles are the Access Grid (AG) and Virtual Rooms Videoconferencing System (VRVS). While both AG and VRVS are based on IETF standards, the bundling of these standards as a single service provides unique capabilities but also makes it difficult to interoperate with standard architectures.

Figure 4.2 illustrates AG's most significant unique features that include multiple video windows per AG node, multiple audio streams, and very large screens so that all views from each participating node can be seen in each location.

Figure 4.3 illustrates the VRVS architecture. The reflectors layer (circled in yellow) is the service core of VRVS. Here, users login to local servers, virtual rooms are hosted along with scheduling calendars, and translations to and from supported protocols are provided.

While H.323 and SIP are based on the concept of one person calling another person (the telephone model), AG and VRVS are designed around the concept of entering a virtual meeting room. In the AG model this is called the "Virtual Venue" and corresponds to a group network address (Class D IP Multicast address). In the VRVS model, this is called a Virtual Room and is accomplished by communication among the distributed set of reflectors - users connect to the service and the service makes sure all room participants receive each others

streams. Persons can only be reached through these virtual locations; furthermore, the virtual locations are in general not permanent. A virtual meeting room or Class D address will be assigned for a particular meeting or series of meetings, and then perhaps reassigned for another purpose. A person never has a permanent videoconferencing address in this design.

To accommodate the need to communicate with users of AG and VRVS systems, H.350.4 describes a generic object class that can be used to describe an AG or VRVS user.

4.1.5 Gateways from Standard to Non-Standard videoconferencing systems

An especially useful feature of VRVS is that it provides gateway services so that H.323 endpoints, SIP user agents and Access Grid Nodes are accessible. For instructions for joining Access Grid meetings through VRVS, consult the VRVS Website (Documentation - Frequently Asked Questions) and also this 'How to Guide'.

4.2 Directory Services

4.2.1 Overview

Directory services can be thought of as databases that are have been optimized for reading information. Directory services are meant to store and organize, in a hierarchical structure, large amounts of related information and to make this information easily accessible through a standard query interface.

4.2.2 Enterprise Directory Services

The term "enterprise directory services" refers to central stores of information about people associated with an institution. This centralized data is usually authoritative, meaning that the set of people responsible for entering data correctly and maintaining its integrity are the only sources for that data. For example, an enterprise's Human Resources department has the responsibility for correctly maintaining information about who is an employee of that enterprise, while the telecommunications department is responsible for assigning telephone numbers, and the computer services department is responsible for assigning an email address. If each of these three departments - Human Resources, telecommunications, and computer services - maintain separate lists, each department winds up duplicating information held in other areas. The inevitable results is that Human Resources has a correct list of names, but out of date or incorrect email and telephone numbers; likewise, telecommunications may not be aware that someone has left the enterprise or changed their last name.

Rather than maintaining three separate lists, the enterprise directory approach builds ONE directory that is used by all departments, gathering information from the authoritative sources. Elimination of redundancy and administrative overhead has tremendous ROI, especially for large organizations. For

example, a centralized directory service provides a single place to enable or disable user accounts and passwords for every service provided by the enterprise.

The administrator hoping to deliver multimedia communications services must either replicate these services or link into the existing processes. Certainly videoconferencing and voice over IP service providers should take advantage of enterprise directories rather than providing customers with "closed box" solutions that require either manual data entry or data replication. Early experiences with multimedia communications have shown that after deployment reaches a critical size, the cost of manual or redundant identity management quickly becomes the greatest cost associated with delivering the service.

4.2.3 LDAP

The Lightweight Directory Access Protocol (LDAP) is a standard describing access to directory services. LDAP was derived from the OSI (Open Source Initiative) Directory Services model X.500 known as "DAP" (Directory Access Protocol). DAP runs over the OSI network protocol stack. LDAP's overall data and namespace model is quite similar to X.500. The major difference is that the LDAP protocol is designed to run directly over the TCP/IP stack, which makes it "lightweight". The current version, LDAP V3 (IETF RFC 3377) , includes important security enhancements.

LDAP is a protocol, not a database. A protocol describes messages used to access certain types of data. It is possible to store data in a variety of backend data stores and use the LDAP protocol as a standardized querying interface. LDAP also provides a data model that standardizes the naming and organization of the data. Finally, LDAP servers are designed to optimize read functions, since the main purpose of this service is to answer queries regarding relatively non-volatile data.

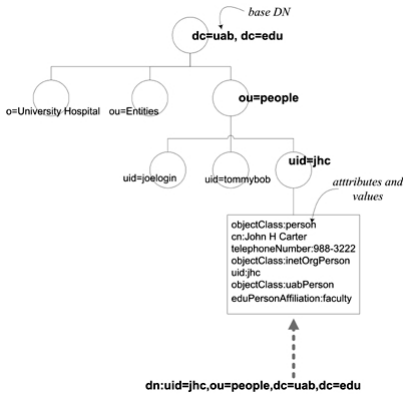
The LDAP information model structures data as a tree - the *Directory Information Tree (DIT)*. An entry in the DIT corresponds to a node in the tree, and contains information about an *object Class*. ObjectClasses have both required and optional *attributes*, and *attribute typing* defines the encoding and matching rules to be used during searching. The LDAP information model is also called the LDAP *schema*.

LDAP provides globally unique naming. By following a path from a node back to the root of the DIT, a unique name is built and is referred to as that node's *distinguished name (DN)*. Figure 4.4 shows an example DIT. Following a path from the gray dotted arrow to the base DN, the unique distinguished name "uid=jhc,ou=people,dc=uab,dc=edu" is built.

Access via the LDAP protocol is implemented by *bindings* (authentication), queries, and updates. Authorization to access data can be managed using *access control lists (ACL's)* which are not standardized.

This very brief summary is intended to provide you with enough vocabulary to read this cookbook. Our resources section at the end contains a number of excellent books and white papers providing more thorough information. A handy website for learning more about LDAP schema is Alan Knowles' LDAP

Figure 9: 4.4



Schema Viewer. Another useful tool is Jarek Gawor's LDAP Browser/Editor software, available from the University of Chicago. This freely available software provides a user-friendly Windows Explorer-like interface to LDAP directories with tightly integrated browsing and editing capabilities. It is entirely written in Java with the help of the JFC (SwingSet) and JNDI class libraries and connects to LDAP v2 and v3 servers.

4.2.4 LDAP Directory Server Implementations

A number of LDAP directory server implementation choices are available. There are a number of factors to consider when selecting your implementation, including price, familiarity with supported platforms, or integration with existing infrastructure.

- OpenLDAP is an open source LDAP Directory Server implementation. Chapter 9 describes how to install and configure OpenLDAP for use with H.350.
- SunOne (iPlanet) Directory Server software began under the name 'iPlanet' (formerly Netscape) and has since moved to 'Sun One' under new management. This is a popular product and is relatively easy to install and configure. Chapter 8 describes how to install and configure SunOne iPlanet for use with H.350
- Active Directory is Microsoft's proprietary directory service for Windows 2000 and 2003. Active Directory supports the LDAP protocol. Chapter 10 describes how to install and configure OpenLDAP for use with H.350.
- Novell Directory Service (eDirectory, formerly called NDS) is Novell's proprietary directory service for Netware. NDS supports the LDAP protocol. We would be delighted to have someone provide instructions for installing and configuring eDirectory for use with H.350.

4.3 Security for Videoconferencing/Voice over IP

Security for videoconferencing requires consideration for both the signaling and media channels. While media transmission is typically end-to-end, the signaling (call setup for H.323, session initiation for SIP) involves a number of hops from endpoint through callservers or gateways to remote endpoint. Ironically, efforts to secure networks such as installing firewalls and network address translators, often interfere with the ability to communicate. An entire book could be written about security and multimedia conferencing and thus is well beyond the scope of this section; the focus here is on two core foundations for security: authentication and authorization, and on security specifications that are already part of existing conferencing protocols.

Authentication is the act of establishing your identity: "Who are you?", in a manner that can be considered to be reliable. Authorization is a decision made by the system regarding what you (the identity you have established) are allowed or not allowed to do. If you cannot reliably establish a person's identity, it is not possible to make meaningful authorization decisions (except for DENY ALL, of course). H.350 has attributes used for authentication and authorization; these attributes were designed with each of H.323 and SIP's existing security protocols in mind, so that no changes to existing standards would be needed to use these security attributes. This design goal is discussed in detail in Chapter 5.

4.3.1 H.323 Security Standard H.235

Security for H.323 is described by ITU-T standard H.235 "Security and encryption for H-Series multimedia terminals". The scope of this standard is to provide authentication, privacy and integrity for H-323. It is important to note that H.323 is generally a device-centric protocol; "the endpoint registers". H.235 provides a means for a person, rather than a device, to be identified. Annex D describes a simple, password-based security profile; Annex E describes a profile using digital certificates and dependent on a fully-deployed public-key infrastructure; and Annex F combines features of both D and E. Use of these security profiles is optional and in the marketplace today, H.323 vendors have left these profiles largely unimplemented in their products.

4.3.2 SIP Security Standard

The scope of SIP security includes authentication, encryption, and non-repudiation via digital signatures. SIP authentication uses a mechanism called SIP digest (due to the use of an MD5 hashing function on the username/password combination). Digital certificates are used after authentication for digitally signing messages, to prevent message tampering.

4.3.3 Does multimedia conferencing really need authentication and authorization?

Each of H.323 and SIP permit use of IPSec (network layer encryption) or TLS (transport layer encryption) between elements in the call path; however, these techniques are considered to be orthogonal to the protocols themselves. You may be wondering at this point why authentication is important to the application at all. Some answers include that billing for a service cannot be done without some confidence that bills are generated correctly and sent to the appropriate person for payment, and that a service provider may wish to offer different service levels at different rates and cannot decide which rate to charge without some form of authentication.

Any of these business drivers could be met with a local server of some type; the challenge is more evident when considering a federated model of service. If public institutions in North Carolina operate a multimedia service for anyone associated with any of those institutions, it is unlikely that there would be a single source for authenticating all those persons. Each university has its own authoritative lists of people and probably does not use identical authentication methods. The state-wide call server may have to execute a decision such as "can this person from UNC-Charlotte contact the classroom unit at UNC-Chapel Hill at this time of day?". The authorization decision - permission to use - is made at Chapel Hill while the user's identity is known to the Charlotte campus. The remote usage policy decision point may instead be in Alabama, or the Netherlands, where an entirely different organization manages the service. Neither H.323 nor SIP today provide solutions to this federated administration model; certainly reliable authentication based on authoritative institutional data is a key part of the solution.

5 Introduction to H.350 Directory Enabled Middleware for Multimedia

Early providers of videoconferencing and voice over IP services found that they could support a few dozen users with manual and marginally secure operational practices. The growing interest in Internet video teleconferencing, instant messaging, voice over IP, data sharing, and other collaborative activities, however, has increased the number of users has into millions. Providers of these services have thus looked for more scalable and manageable architectures order to reliably and securely meet their users' expectations.

5.1 Recommendation H.350

Current multimedia communications technology on the market today typically is not designed to integrate into enterprise identity management processes. However, the International Telecommunication Union's (ITU) recently completed H.350 series of Recommendations provides a standardized way to manage mul-

timedia communication information in the enterprise. H.350 describes a standardized schema that represents multimedia conferencing information in LDAP directories, including SIP, H.323, H.235 and H.320 protocols. Non-standard protocols can also be represented, which is useful for facililating communication with "virtual room" based conferencing technologies such as Access Grid and VRVS. The H.350 schema is designed to require minimal changes to the enterprise directory.

5.2 Design Goals

H.350 was designed with the following goals:

5.2.1 Authoritative Data Source for SIP, H.323, H.235, H.320 and proprietary call servers

Two directories are involved: the enterprise person directory, and the H.350 directory. The enterprise person directory contains at least the inetOrgperson object class (and perhaps the eduPerson object class) which include name, address, and various methods for contacting that person such as telephone, fax, email, and commURI. The commURI is simply an LDAP URI that points to a user's multimedia conferencing information in the H.350 directory. The URI has a label to give a friendly name to each endpoint, such as 'Office Telephone' or 'Home Instant Messaging.' Each instance of a communication object will have a commURI and it will be common for users to have multiples. This approach is relatively leverages use of the existing enterprise directory and is relatively non-invasive to the enterprise directory: all that is required is a commURI.

The data for each multimedia conferencing endpoint is contained in the H.350 directory in the entry pointed to by a user's commURI. This includes all of the protocol elements and addresses for that endpoint. Each instance of a multimedia communications endpoint is represented by a single entry in the directory. Each instance may have multiple addresses associated with it or even multiple protocols, if the associated device supports those protocols. A pointer, called 'commOwner', refers back to the user with whom this endpoint is associated. Thus, given a user, it is possible to find all of her endpoints. Given an endpoint it is also possible to find its user. No other copy of this data is maintained. It is separate from the enterprise directory so that its operation can be optimized for multimedia communications. It may also be convenient to partition the management of H.350 services from enterprise directory services, especially where two sets of managers are involved.

<COULD USE SOME DIAGRAMS HERE>

5.2.2 Automated Client Configuration

SIP proxies and H.323 gatekeepers are examples of call servers that can perform registration and call routing for multimedia communication services. In some

cases, these call servers will be H.350 aware, such as the RADVISION ECS system. These H.350 enabled call servers can pull their information directly from the H.350 directory without further modification. This is the simplest and best approach to directory enabled conferencing. As an additional benefit, endpoints can be properly configured by pulling information down from the H.350 directory, which solves the problem of manually configuring endpoint software at the desktop. By linking account management and authorization automation to the enterprise directory using the LDAP protocol, H.350 enables companies and universities to scale up video and VoIP operations from a few hundred endpoints to full enterprise deployments without hiring additional systems administrators.

Many systems will not be H.350 aware, and instead will utilize some proprietary data management scheme, such as an internal database or proprietary LDAP schema. In these cases the authoritative H.350 information may be replicated and stored in the call server's expected data store and format.

5.2.3 White Pages Listings and Lookups

H.350 allows you to search for and find a user's video or VoIP address just like you would find an email address or telephone number today. Because it is standardized, H.350 directory listings will work with multiple vendors' equipment. An additional benefit of standardization is that "Directory of Directories" or metadirectory searches are possible, resulting in the potential for a global videoconferencing directory.

5.2.4 Support for 'clickable' dialing where appropriate.

<NEED SOME WORDS HERE>

5.2.5 Authentication

H.350 provides a storage location for authentication credentials that is both convenient and consistent. One of the barriors to PKI deployment has been the indeterminant location of the required credentials; H.350 provides a consistent and standardized location for storing digital certificates and other credentials such as passwords used by the supported protocols. Thus, video and VoIP conferencing can enjoy a more secure operational environment using H.350. A more detailed discussion of H.350 support for authentication and the relationship of these credentials to enterprise credentials follows later in this chapter.

5.2.6 Authorization

H.350 contains a service provider defined "class of service level" attribute. Depending on the local usage policy, the attribute could be populated based on type of equipment (the commObject attributes stored in the H.350 directory) or the authorization attribute could be populated based on information in the person directory ("executive level", "student", or "gold level customer"). This

information can be used by the call server to permit or prioritize access to services.

5.3 Overview Leveraging Enterprise Authentication

In this section, the question of how the endpoint and call server learn the needed credentials is discussed. This section discusses only H323 authentication scenarios, although similar issues exist in SIP.

Three potential data storage scenarios exist for endpoint and gatekeeper configuration information:

- Non-Standard Storage : No use of h235Identity commObjects
- Endpoint-only use of H.350 Storage
- Endpoint and Gatekeeper use of H.350 Storage

5.3.1 Non-Standard Storage

< Figure: Non-Standard Storage>

Prior to the adoption of H.350, there was no standard way to store EndpointID or Password. The endpoint and gatekeeper have separate data stores for this information. The endpoint typically "reads" data, like passwords, when entered by the user. The gatekeeper has some predefined list of users stored in a database it uses.

5.3.2 Endpoint-only Use of H.350 Storage

< Figure: Endpoint h323Identity-based Storage >

When commObject is introduced into the configuration, the endpoint is able to read its configuration data from a well known location: the h323Identity and h235Identity objects on the H.350 server. In this scenario, the gatekeeper is still using its own private data store to define users and passwords.

5.3.3 Endpoint and Gatekeeper Use of H.350 Storage

< Figure: Endpoint and Gatekeeper h323Identity-based Storage >

The third scenario, Figure , shows a configuration where both the endpoint and the gatekeeper use the same h323Identity commObject for storing information about a particular endpoint. The endpoint and gatekeeper have different uses for this object. The endpoint uses the object to completely configure itself in preparation for H323 activity. The gatekeeper, however, uses the object as its database of videoconferencing-specific user names and passwords. If the gatekeeper and endpoint both read the user's h235IdentityEndpointID and h235IdentityPassword from the same database, they will naturally be using shared values for authentication exchanges.

There are a few issues with the third scenario. The endpoint and gatekeeper need different types of access the data. For example, an endpoint should only

be able to read its own configuration object and no one else's, otherwise the passwords would not be secret. A gatekeeper, however, needs to read at least the h235IdentityEndpointID and h235IdentityPassword for every endpoint that will attempt to authenticate. This can be handled, for example, by ACL rules in LDAP. A straightforward way to restrict access correctly is to have the endpoint bind to the LDAP server with the users credentials, and use an associated ACL.. The gatekeeper, however, can access the commObject LDAP server with a privileged account that has been given access to all commObjects.

Once the endpoints and the gatekeeper are using the same database, they are effectively synchronized with each other. While this scenario provides a way to synchronize authentication credentials between the endpoint and gatekeeper, it does not in itself address enterprise authentication.

5.3.4 Enterprise Identity

An enterprise user typically has some enterprise identity (EntID) and associated password that can be used to access services across the enterprise. It's desirable for enterprise users to access video conferencing and IP Telephony services using the same EntID and password. This is easy for the end-user, but introduces the problem of protecting those credentials from misuse. If they are compromised, then all of the services available to the user across the enterprise would also be compromised.

5.3.5 Enterprise Authentication using LDAP

In many environments, LDAP has grown from being just the place where white-pages information is stored to a source for central authentication credentials. Because LDAP is a common authentication mechanism, it's helpful to review the authentication methods that can be used when connecting (binding) to an LDAP V3 server.

Clear Text Password This type of bind sends an unencrypted user name and password to the LDAP server for verification, and can be secured only by using a secure channel to transmit the password, eg. SSL.

Secure Authentication and Security Layer (SASL) This type of bind allows the client and server to negotiate an authentication mechanism of their liking. Supporte mechanisms include Kerberos, SKEY, external, or other authentication mechanisms.

Digest MD5 Hashed Password This type of bind secures the authentication credentials during a bind by sending hashes of data based on the password. It is a form of SASL authentication and is similar to how H.323 protects the password during its authentication.

Client Certificate This type of bind uses client certificates at the SSL layer to identify the user. It is a form of external authentication using SASL.

What's important to remember about all of these LDAP authentication methods is that they assume that the authentication is happening just after the user has entered their password at a prompt; in other words, in the authentication exchange the LDAP client learns the clear-text secret that is used to negotiate the authentication. The LDAP server will use its internal database to find the secret corresponding to the client user in order to verify that the user has the correct secret. (Note: this is true even for the certificate based authentication. While the LDAP server only needs a public key, the LDAP client must have the private key).

5.3.6 Enterprise Credential Access

In an enterprise environment with a single set of administrators, one can imagine that the EntID and password could be used directly in an H323 authentication sequence. The endpoint would prompt the user for their EntID and password and then use those credentials to negotiate the authentication exchange with the gatekeeper. The gatekeeper would access the enterprise's store of EntIDs and passwords when an authentication request arrives, retrieve the corresponding user's credentials, and then be able to verify that the user at the endpoint possess valid enterprise credentials. This solution assumes that the H323 services are run by the same organization that manages the enterprise credentials and that it can be ensured that all authentication components will have access to the user's enterprise credentials. This solution roughly corresponds to Figure .

In heterogeneous enterprise environments, where the enterprise authentication services are frequently managed by one organization and the H323 services are managed by another, there are not only differences in usage and access policies for the services there may be significant differences in the technologies used to store data. We're aware of these issues, and this is why we are migrating services to use standard data access protocols like LDAP. Furthermore, the enterprise authentication service providers may not be able to accommodate requests to access a user's enterprise credentials for each service that wants access, either because of political boundaries or trust barriers. It's not unreasonable to be cautious about handing out access to user credentials. These realities frequently lead to each system environment defining it's own set of credentials for users. An enticing solution to this problem is to allow each system environment to use their own authentication credentials but tie access to those credentials to a user's enterprise credentials.

5.3.7 Enterprise Authentication at the Endpoint

Endpoints that are upgraded to use h323Identity commObjects will inherently be LDAP aware. In order for an endpoint to read the h323Identity, it needs to bind to the LDAP server that contains commObjects. Enterprise authentication credentials can be tied in at this point by requiring the endpoint to use an authenticated bind to the commObject server with the user's enterprise credentials. After a successful bind, the endpoint will know that this is a valid

enterprise user and can have access to only that user's h323Identity commObjects. While this neatly ties in enterprise authentication at the endpoint, it doesn't describe how those enterprise credentials could be used by the H323 authentication exchanges.

5.3.8 H.323 Authentication Options

There are two options for H323 authentication at this point: use the enterprise credentials or use the information stored in the h235Identity credential attributes.

The user's enterprise credentials can only be used directly for H323 authentication in environments where the gatekeeper has access to the user's enterprise credentials. This is because both the endpoint (client) and the gatekeeper (server) need know the user's secret so they can challenge each other to verify that they know it. The secret never travels from the endpoint to the gatekeeper. In other words, the gatekeeper will never "get" the secret from the endpoint.

It is not possible for the gatekeeper to simply "participate" in the enterprise authentication scheme like the endpoint did in the LDAP bind scenario above because the gatekeeper can not verify the user's password by trying to perform an authentication with that password against the enterprise authentication system. In order to validate the user in this way, the gatekeeper would need the user's clear-text password, which is not permitted in this scenario.

The second option is to use the enterprise credentials of the user to "unlock" the h235Identity data. This is the scenario of Section . In this scenario, both the endpoint and the gatekeeper will use the h235Identity object associated with the current endpoint. This scenario is represented by Figure . The endpoint will access the h323Identity in order to retrieve it's configuration including the h235IdentityID/ h235IdentityPassword or userCertificate, as needed. The advantage of this approach is that the endpoint and gatekeeper will be using the same data store so they will naturally be using the same secret to negotiate the authentication. The endpoint can access the commObject LDAP server with the user's enterprise credentials and the gatekeeper can access commObject LDAP server with it's own credentials. The end-user is completely unaware of the fact that a separate password is being using to negotiate the H323 authentication. All the user did was present their enterprise authentication. This approach also offers some autonomy between the H323 system and the enterprise system.

The biggest drawback to this approach is the management of the h235Identity credentials. How does one set the credentials? Should they be permanent? If they change, how frequently should they change?

5.3.9 Dynamic Credentials

Dynamic credentials could be used to address the questions and concerns over managing the h235Identity credentials and (potentially) sending them over unencrypted communication channels to the endpoint. If this password could be changed on a regular basis, then security might be improved. Dynamic cre-

dentials aren't a requirement. It's conceivable that the credentials could be set when the h323Identity commObject is defined and then never be changed again. The solution of having the endpoint and gatekeeper both access the h323Identity would work just the same. The reason dynamic credentials are desirable is because permanent passwords are vulnerable to security breaches.

This is especially true in environments where a clear-text piece of data is stored in an LDAP object that might be read using an insecure channel. Keep in mind that even if secure authentication is performed, LDAP does not inherently protect the data that is transfered afterwords. Only if a secure transport is used or if security is negotiated for the connection will the data be kept secure as it travels the network. Since the h235Identity credentials are transparent to the end user, their change policy can also be transparent. The enterprise user just needs to remember their enterprise credentials. The H323 system will ensure that the correct H323 credentials are used internally. There are a few approaches to changing the h235Identity credentials. If they are short-lived one could potentially build a Kerberos-like system where the authentication credentials have a life span just long enough to negotiate the H323 authentication but be useless for later attempts. The problems with these approaches are an agreement on how long the life time of the credential should be and where the life time is defined. Three options are considered below in order of increasing desirability.

Change h235IdentityPassword Regularly

The simplest option is to just change the h235IdentityPassword on some regular basis. It might be most convenient to do this only when the h323Identity is read by and endpoint. Some LDAP servers internally support taking specific actions when a user binds to the directory. This mechanism could be used to set the h235IdentityPassword whenever a user binds to the commObject server. One problem with this approach is that once the password is set it wouldn't expire. It would remain the same until the user binds to the commObject server again. This might not happen for a long time. It would be nice to expire the password. The problem is how long should the life time of the password last and how do you share that time period with the endpoint and gatekeeper so they know when not to use the password anymore? Unfortunately, the password itself won't contain any timing data, so some type of outside timing source would be required (although a temporary digital certificate COULD carry a timestamp).

Change h235IdentityID and h235IdentityPassword Regularly

The second approach would be to use some common time stamp to show when the password expires. The time stamp could then travel with the password and the password could always be checked against it. The h235IdentityID might suffice as a time stamp (ie. generate an h235IdentityID based on the expiration time every time you generate an h235IdentityPassword), but this might not be a legitimate use of the attribute nor a very expected one. The mechanism still requires that the clocks between the password/uid(timestamp) generator

and the consumers be synchronized. It also requires that the endpoints and gatekeepers be updated to agree on this use of h235IdentityID. It also seems fairly fragile.

Use short-lived userCertificate in h235Identity

The third approach is to use short lived certificates, similar to the ones generated by the pkcs11 component in the kx5093 world. This approach is appealing because certs have a clearly defined expiration mechanism, eliminating the UID hack of the previous example. The client knows when the certificate expires and can just re-read it's h235Identity to get a new one once the old one expires. The gatekeeper has the same advantage. It can simply throw out expired certs and if a new authentication request arrives from the endpoint, it can look up the cert again. Another advantage of this approach is it fits in nicely with the certificate based infrastructure of H323.

5.3.10 SIP and Enterprise Authentication

5.3.11 Leveraging Enterprise LDAP Authentication

5.3.12 Additional approaches to integration with enterprise authentication

6 Architectural Decisions for Implementers

6.1 Directory Architectural Decisions

There are two alternative implementations when building the directory services. The Enterprise Directory and *commObject* Directory are intended to be viewed as logical units and not physical directories. They can both be considered directory branches and can be implemented as one central directory server or two physically different directory servers. The advantages and disadvantages of the two structural choices are described below.

6.1.1 Person Directory and *commObject* Directory

commObject information is located separately from person or resource information. Its location may be a sub-tree of the larger enterprise directory or on a separate logical server. The person directory will continue to host traditional person or resource information such as name, telephone, address, etc. In addition, it will contain a commURI link to the commUniqueId attribute in *commObject*. Rather than extending the enterprise directory's person object class, this linking provides the following advantages:

1. Changes to the enterprise directory are not to be undertaken lightly and are often not under the administrative control of the video/voice over IP service provider.

2. Elements associated with video and voice over IP communications are very dynamic. The technology itself is changing quickly in relation to the enterprise directory. Separation allows changes to the *commObject* LDAP infrastructure without modifying object classes of the enterprise directory.

3. A call server may need to access *commObject* data very differently than other applications access the enterprise directory. A separate server can be tuned for performance and access policy to accommodate these implementation requirements. For example, a call server may need to query the *commObject* server many times per second in order to handle real-time call processing, or it may read and cache many *commObject* attributes at once.

Any user or resource with multimedia conferencing capabilities should have an instance of *commObject* created and linked to an existing entry in the enterprise directory with a commURI.

6.1.2 Single Directory Server

The Single Directory Server approach is best to employ when simplicity is the driving factor and no enterprise directory service exists. A single directory server is recommended when services will be provided for only a small number of people. A distinct advantage of implementing both directory branches in a single directory is that everything is in one location. With the Enterprise Directory and *commObject* Directory branches in the same directory, administration is limited to that single directory and there is no need to worry about communication between two different directories.

With this approach, some thought must be given to how you will manage user accounts. If your server is "open to the public", you may choose to allow anyone to create an account, set their identity, and set and reset their own password. If "do it yourself" account management does not fit your requirements – for example, only people who are current employees may use the system – you will need solutions for verifying user identity and status, and should be prepared with a solution to integrate the information you have with an enterprise printed or on-line directory.

Registration services require privileged access to the *commObject* Directory branch because the call server must verify that information provided by the user matches that stored in the directory before registering the user. It is possible to allow privileged access to only one branch in the directory; however, because of the complexity and lack of standardization of LDAP access control rules (ACI rules), it is too easy for novice LDAP managers to err in setting access control. One mistake in the ACI rules could give the registration server (and perhaps a hacker) access to all of the information in the Enterprise Directory branch.

Another disadvantage to the single server approach is that all queries will be made to this single directory server. Placing both branches into the same server doubles the load on a single directory server.

6.1.3 Separate Enterprise Directory and *commObject* Directory Servers

It is best to implement the Enterprise and *commObject* Directory branches in two different directories when load and security on the Enterprise Directory are the major concerns. It is also easier to implement the architecture in this way when there is a preexisting Enterprise Directory already in place. Most large universities already have enterprise-level people directories, and it is an advantage to have this in place. The enterprise directory managers may not be ready and willing to manage the entire *commObject* schema, especially given the need for privileged access to the directory. With a separate *commObject* directory, only minimal changes need to be made to the Enterprise Directory in order to support H.350.

Using separate directories reduces the number of queries on one single directory. By balancing the query load on different directories, the traffic on the Enterprise Directory is greatly reduced. Another advantage to using separate directories is that by separating the Enterprise Directory and *commObject* Directory onto two distinct servers, , it is much easier to provide tight security on the Enterprise Directory while still allowing privileged access to the *commObject* Directory..

One of the disadvantages of this approach is that there are additional BIND requests needed. This isn't a terrible burden overall, but it is additional overhead which should be discussed. With the standalone central directory, you need only BIND to one directory. All subsequent queries are made to that directory. For this scenario, however, there are two directories with which you must BIND.

Another disadvantage of having separate directories is simply the increase in complexity. Adding additional components to any architecture will obviously increase the complexity of the architecture. With the addition of another directory, an additional directory administrator may be needed. Communication between the directory service managers will need to be organized and pass-through authentication will need to be established.

6.1.4 Coordinating Enterprise and Videoconferencing Directory Services

If different groups manage the enterprise and H.350 directories, what methods are suggested for coordinating data entry between the two?

6.1.5 Public Directories

Multimedia communication by definition is about connecting people that are far away. People within an institution typically want to utilize multimedia communications in order to collaborate with users at other institutions. For mature technologies such as electronic mail, it can be assumed that those remote users will have access to their own services. However new technologies such as multimedia communications require that the institution make provisions for some level of service both to its own internal users as well as the individuals that are outside of the institutional purview but with whom there is a legitimate interest

in collaboration. For this reason, it is important to provide a basic level of directory service. This will typically consist of the ability of users to create their own LDAP inetOrgPerson (or eduPerson) identity to store name, address, telephone, email and related basic information. It will be configured to automatically notify users every 180 days to refresh their subscription, or else be purged out of the system. It will provide a way for users to reset forgotten passwords. Early pilots using this approach in ViDeNet have yielded very positive and scalable results. In general, the assumption is that there will be external users, and it is better for them to be managed in some way, rather than not managed at all. As adoption of multimedia communications becomes widespread, these public services can be phased out.

7 Directory Schema

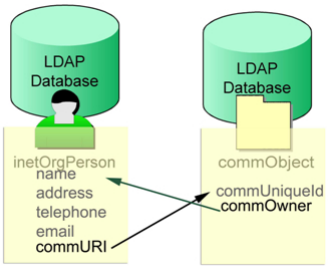
7.1 Person/Resource Schema

For directory services that are useful for videoconferencing and voice over IP, an association must be created between a person and a device or communication service. When you look up a person in a directory, you'd like to be able to discover their email address, telephone number (for analog and/or Ethernet phones) and also how to reach them via videoconferencing. H.350 provides an association of person to endpoint using the commURI object class, which contains a labeled URI pointing to device or protocol specific information for each type of videoconferencing endpoint or service used.

The *commObject* directory contains protocol-specific information. It is important to create an association between a device and the person who is using it - perhaps for billing purposes or to establish classes of service based on attributes stored in an eduPerson directory such as "student" or "faculty". H.350 provides an association of an endpoint with its owner using the *commObject* object class, which contains a labeled URI named commOwner pointing to the person directory entry of its owner.

Some multimedia conferencing implementations are heavily endpoint-oriented, whereas others are user-oriented. A group video teleconferencing endpoint in a conference room, for example, may be referred to as 'Conference Room 201'. This endpoint is not associated with any particular person, but is associated with the resource Conference Room 201 and is shared by whoever needs to use the conference room. Other endpoints may be user context-specific, deriving their identities from the current user. For example, when logging onto a computer as jdoe, a computer-based endpoint may configure itself with the address of jdoe as stored in that user's profile and register with a call server accordingly. Other users logging onto the same computer may have different identities and different user profiles so their registration messages from the one computer will contain different identity information.

Figure 10: 7.1



This dichotomy of users versus resources makes it difficult to associate endpoints with users or resources. Linking *commObject* to a person via a commURI generalizes this relationship. If a commOwner attribute is pointing to a person object class, then that *commObject* is associated with that person. If a commOwner attribute is pointing to a resource object class, then that *commObject* is associated with that resource. Both people and resources can have commURI pointers that associate endpoints with a directory entry. Enterprise directories that support only people and not resources may choose to simply treat resources as people.

7.1.1 inetOrgperson

The inetOrgPerson object class is a general purpose object class that holds attributes about people. The full specification for inetOrgPerson can be found in the IETF RFC 2798 "The Definition of the inetOrgPerson LDAP Object Class" .

The following inetOrgPerson example is provided in LDIF format.

version: 1
dn: cn=Theresa Miller,ou=uab,dc=uab,dc=edu
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Theresa Miller
cn: Terry Miller
displayName: Terry Miller
sn: Miller
givenName: Theresa
initials: TRM
title: associate director
uid: tmiller
mail: tmiller@uab.edu
telephoneNumber: +1 934 555 1234

facsimileTelephoneNumber: +1 934 555 9999
mobile: +1 934 555 1941
roomNumber: 510
carLicense: UAB13
departmentNumber: 2604
employeeNumber: 1864291
employeeType: faculty
labeledURI: http://www.uab.edu/users/tmiller/myhomepage.html

The inetOrgPerson object class is commonly included with OpenLDAP and iPlanet distributions.

7.1.2 eduPerson

EduPerson is an auxiliary object class for campus directories designed to facilitate communication among higher education institutions. It consists of a set data elements, or attributes, about individuals within higher education, along with recommendations on the syntax and semantics of the data that may be assigned to those attributes. If widespread agreement and implementation of this object class in campus directories is achieved, a broad and powerful new class of higher education applications can be deployed. Additional information on eduPerson is available at its home on the web: http://www.educause.edu/eduperson.

It is important for a directory to have the Person, orgPerson, and inetOrgPerson object class hierarchy in place in addition to eduPerson. Ultimately, attributes are what are important, but object classes (both eduPerson and its parent classes) give additional confidence that the semantics of the attributes are consistent between sites. Also, other applications will have expectations that the parent object classes exist. Note that even with the full hierarchy in place there is no requirement to populate any of the non-mandatory attributes in the parent object classes.

A number of higher education groups (Internet2, NET@EDU, and Educause) are working to make eduPerson a voluntary "standard" for describing university people. The eduPerson object class extends the inetOrgPerson class with attributes supporting inter-campus access to shared resources.

7.1.3 Resource Object classes

Implementers who are interested in listing resources as well as people in your directory (for example: "Distance Learning Classroom #3") may be interested in using a standardized resource object class. IETF RFC 1274 "Cosine and Internet X.500 schema" includes definition of a resource object class.

The Room object class can be used to represent rooms:

objectclass (0.9.2342.19200300.100.4.7 NAME 'room'
SUP top STRUCTURAL
MUST commonName

MAY (roomNumber $ description $ seeAlso $ telephoneNumber
))

The device object class can be used to represent devices:

objectclass (2.5.6.14 NAME 'device'
DESC 'RFC2256: a device'
SUP top STRUCTURAL
MUST cn
MAY (serialNumber $ seeAlso $ owner $ ou $ o $ l $ description
))

7.2 H.350 Schema

H.350 is the base document for the series. It uses a generic object class called *commObject* to represent attributes common to any video or voice protocol. Auxiliary classes represent specific protocols such as h.323, h.235, sip, etc. Multiple H.350.X classes can be combined to represent endpoints that support more than one protocol. For example, endpoints that support H.323, H.235 and H.320 could include H.350, H.350.1, H.350.2, and H.350.3 in their LDAP representations. In addition, the entry should contain commObject to serve as the entry's structural object class.

There are two basic components in the architecture:

- The *commURI* object class is a class whose only purpose is to link a person or resource to a *commObject*. By placing a *commURI* 'pointer' in an individual's directory entry, that individual becomes associated with the particular targeted *commObject*.
- The *commObject* object class contains a pointer called *commOwner* that points to the individual or resource associated with the *commObject*.

In this way, people or resources can be associated with endpoints. The only change required in the enterprise directory is the addition of the simple object class *commURI*.

CommObject data may be instantiated in the same directory holding the person information, or in an entirely separate directory, thus allowing flexibility in implementations.

7.2.1 Review of H.350 Design Goals

Large-scale deployments of IP video and voice services have demonstrated the need for complementary directory services middleware. Service administrators need call servers that are aware of enterprise directories to avoid duplication of account management processes. Users need 'white pages' to locate other

users with whom they wish to communicate. All of these processes should pull their information from canonical data sources in order to reduce redundant administrative processes and ensure information accuracy. Toward that end, H.350 was designed to:

1. Enable endpoint information to be associated with people or with resources such as conference rooms or classrooms.

2. Enable online searchable "white pages" so that dialing information (e.g. endpoint addresses) can be found along with other "traditional" directory information about a user, such as name, address, telephone, email, etc.

3. Enable all endpoint information to be stored in a canonical data source (the Directory), rather than local to the call server, so that endpoints can be managed through manipulations of an enterprise directory, rather than by direct entry into the call server.

4. Support the creation of very large-scale distributed directories such as white pages "portals" that allow searching for users across multiple institutional directories. Standardized directory object classes enable searches across multiple LDAP directories for multimedia-specific information.

5. Support multiple instances of endpoints per user or resource.

6. Represent endpoints that support more than one protocol, for example endpoints that are both H.320 and H.323.

7. Store enough information about endpoint configuration so that correct configuration settings can be documented to end users on a per-endpoint basis, as a support tool, or loaded automatically into the endpoint.

8. Be extendable as necessary to allow implementation specific attributes to be included.

9. Be non-invasive to the enterprise directory, so that support for multimedia conferencing can be added in a modular fashion without significant changes to the enterprise directory.

7.2.2 Extending the H.350 schema with Auxiliary classes

H.350 object classes may be extended as necessary for specific implementations. For example, a class may be extended to support billing reference codes. Developers that implement proprietary endpoint functionality may need that functionality to be represented by attributes in the directory, or an enterprise may want to use attributes such as 'modelNumber', and 'accountNumber' that are not defined in the standard but may be useful if implemented. Extensions to the schema are not considered to be part of the standard.

A full discussion of schema design and extension is beyond the scope of this document. See IETF RFC 2252 for details. Most importantly, adding attributes

to this architecture must be done in a way that does not break compatibility with the H.350 standard. The only method we recommend be used to extend H.350 is by defining an auxiliary class. The auxiliary class will should have the special class top as its superior. The following example creates billing account and billing manager attributes by defining them in their own auxiliary class.

```
objectclass ( BillingInfo-OID
NAME 'BillingInfo'
DESC 'Billing Reference Information'
SUP top AUXILIARY
MAY ( BillingAccount $ BillingManager $ )
)
```

We recommend that all attributes in the auxiliary class be optional rather than mandatory. This way the auxiliary object class itself can be associated with an entry whether there are any attribute values for it.

The following example shows a sample endpoint that utilizes the new auxiliary class and attributes. This example also uses H.350.1 for h323Identity.

```
dn: commUniqueId=2000,ou=h323identity,dc=company,dc=com
objectclass: top
objectclass: commObject
objectclass: BillingInfo
commUniqueId: 2000
BillingAccount: 0023456
BillingManager: John Smith
```

7.2.3 Attribute Object Identifiers (OID)

Each attribute has an Object Identifier (OID),a globally unique numerical identifier usually written as a sequence of integers separated by dots. For example, the OID for the *commUniqueId* is 0.0.8.350.1.1.2.1.1. All attributes must have an OID. *CommObject* OID's have been assigned by the ITU. The LDIF files we provide at the end of the next chapter contain correct object classes, attributes and OIDs needed to implement H.350.

7.2.4 Indexing

An important part of LDAP server administration is deciding which attributes to index. Non-indexed attributes can result in search times sufficiently long as to render some applications unusable. As in any database design, attribute indexing decisions are made with implementation-specific activities in mind. Certainly, user (dn) and alias lookup should be fast. The next chapter includes an Indexing Profile for *commObject* directories that is optimized for use in directory of directories applications. Use of this profile is optional.

8 Object Class and Attribute Definitions and Examples

8.1 commURIObject (H.350)

OID: 0.0.8.350.1.1.1.2.1
objectclasses: (0.0.8.350.1.1.1.2.1
NAME 'commURIObject'
DESC 'object that contains the URI attribute type'
SUP top AUXILIARY
MAY (commURI)
)

Definition & Use: Auxiliary object class that contains the commURI attribute. This attribute is added to a person or resource object to associate one or more commObject instances with that person/resource. commURIObjct values are LDAP URIs pointing to the associated commObject representing, for example, a user's H.323 conferencing station and SIP IP phone. Multiple instances of commURI need not point to the same commObject directory. Each commURI instance could point to an endpoint managed by a different service provider.

8.1.1 commURI

OID: 0.0.8.350.1.1.1.1.1
attributetypes:(0.0.8.350.1.1.1.1.1
NAME 'commURI'
DESC 'Labeled URI format to point to the distinguished name of the commUniqueId'
EQUALITY caseExactMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

Definition & Use: Labeled URI containing an LDAP URL identifying the directory containing the referenced commObject instance. The search filter specified by this LDAP URL shall specify an equality search of the commUniqueId attribute of the commObject class. Used to find the endpoint of the user in question. The label field may be used to represent the function of the endpoint, such as 'home IP phone' or 'desktop video' for user interface display purposes. The label portion of the field may contain spaces as in the example below showing 'desktop video'.

Number of Values: multi

Indexing Profile: No recommendation

Example (LDIF fragment):
commURI: ldap://dir.acme.com/dc=acme,dc=com??sub?(commUniqueId=bob) desktop video

8.2 commObject (H.350)

OID: 0.0.8.350.1.1.2.2.1
objectclasses: (0.0.8.350.1.1.2.2.1
NAME 'commObject'
DESC 'object that contains the Communication attributes'
SUP top STRUCTURAL
MUST commUniqueId
MAY (commOwner $ commPrivate)
)

Definition & Use: Abstraction of video or voice over IP device. The commObject class permits an endpoint (H.323 endpoint or SIP user agent or other protocol endpoint) and all their aliases to be represented by a single entry in a directory. Each directory entry should contain commObject as the entry's structural object class; that entry may also contain H.350.X auxiliary classes.

8.2.1 commUniqueId

OID: 0.0.8.350.1.1.2.1.1
attributetypes: (0.0.8.350.1.1.2.1.1
NAME 'commUniqueId'
DESC 'To hold the endpoints unique Id'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

Definition & Use: The endpoint's unique ID. This is the real distinguished name (RDN)of this object. In practice, there will always be one and only one commUniqueId for every endpoint. This attribute uniquely identifies an endpoint in the commObject directory. It must be unique within that directory, but need not be unique globally. This attribute has no relationship to the enterprise directory; it may however be handy to use the enterprise distinguished name as part of commUniqueId..

Number of Values: multi

Indexing Profile: equality

Examples (LDIF fragments): commUniqueId: Device4321
commUniqueId: jilltaylor003

8.2.2 commOwner

OID: 0.0.8.350.1.1.2.1.2
attributetypes: 0.0.8.350.1.1.2.1.2
NAME 'commOwner'

DESC 'Labeled URI to point back to the original owner'
EQUALITY caseExactMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

Definition & Use:

Labelled URI format to point back to the person or resource object associated with this entry. Used as a reverse entry finder of the owner(s). This attribute may point to groups. Note that this URI can point to a cn, but applications that want to bind authentication information across both the commObject and enterprise directories should points to a dn rather than a cn, thus uniquely identifying the owner of the commObject.

Number of Values: multi

Indexing Profile: presence

Example (LDIF fragment):
commOwner: ldap://dir.acme.com/dc=acme,dc=com??sub?(dn=rsmith42)
commOwner: uid=rsmith42,ou=people,dc=acme,dc=com

8.2.3 commPrivate

OID: 0.0.8.350.1.1.2.1.3
attributetypes: (0.0.8.350.1.1.2.1.3
NAME 'commPrivate'
DESC 'To decide whether the entry is visible to world or not'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

Definition & Use: To be used by the user and indicate privacy options for an endpoint, i.e. unlisted number. This attribute is defined as Boolean. Future version of this Recommendation may develop a controlled vocabulary for this attribute to accommodate multiple types of privacy.

Number of Values: multi

Indexing Profile: presence

Example (LDIF fragment): commPrivate: true

8.3 h323Identity (H.350.1)

OID: 0.0.8.350.1.1.3.2.1
objectclasses: (0.0.8.350.1.1.3.2.1
NAME 'h323Identity'
DESC 'h323Identity object'
SUP top AUXILIARY
MAY (h323IdentityGKDomain $ h323Identityh323-ID $
h323IdentitydialedDigits $ h323Identityemail-ID $

h323IdentityURL-ID $ h323IdentitytransportID $
h323IdentitypartyNumber $ h323IdentitymobileUIM $
h323IdentityEndpointType $ h323IdentityServiceLevel)
)

Definition & Use: The h323Identity object class represents H.323 endpoints. It is an auxiliary class and is derived from the commObject class defined in H.350. Implementers should review H.350 in detail before proceeding with H.350.1. Its attributes include all H.323 Alias types. These aliases can be downloaded to an endpoint for automatic configuration, accessed by a gatekeeper for call signaling and authorization, and published to white pages to create user dialing directories. Note that the following seven alias types are defined in H.323 as dialing methods. Each of these alias types is represented below with corresponding h323Identity attributes. Keep in mind that these attributes are separate from information in the enterprise directory. For example, email-ID is a separate field than a user's email address as represented in the enterprise directory. For implementation purposes an administrator may set these values equal by direct entry or by referral.

H.323 dialing method	h323Identity attribute
h323-ID	h323Identityh323-ID
dialedDigits	h323IdentitydialedDigits
email-ID	h323Identityemail-ID
URL-IF	h323IdentityURL-ID
transportID	h323Identitytransport-ID
partyNumber	h323IdentitypartyNumber
mobileUIM	h323IdentitymobileUIM

8.3.1 h323IdentityGKDomain

OID: 0.0.8.350.1.1.3.1.1
attributetypes: (0.0.8.350.1.1.3.1.1
NAME 'h323IdentityGKDomain'
DESC 'FQDN of the Gatekeeper'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

Definition & Use: Specifies the FQDN name or IP address of the gatekeeper to which the endpoint should register. Where endpoint gatekeeper location is configured via H323 URL, please note that this attribute will not hold an H.323 URL with a scheme name but will hold a valid DNS domain name. If an endpoint is provisioned for its Gatekeeper location with only a valid DNS domain name it is assumed that this DNS domain name is the value of the hostport of the H.323 URL. H.323 Annex O section O.8.2 describes this special case. In particular, the endpoint will attempt to

retrieve from the specified domain name value an SRV record indicating the gatekeeper(s) address. If the SRV lookup fails, then the endpoint will attempt to retrieve an A record. H.323 Annex O describes the flow of the lookup process in section O.9.

Number of Values: multi

Indexing Profile: No recommendation

Example Application Using Attribute: A web page that displays a user's proper endpoint configuration information.

Example (LDIF fragment): h323IdentityGKDomain: gk.radvision.com // FQDN example
h323IdentityGKDomain: 1.1.1.1 // IP address example

8.3.2 h323Identityh323-ID

OID: 0.0.8.350.1.1.3.1.2
attributetypes: (0.0.8.350.1.1.3.1.2
NAME 'h323Identityh323-ID'
DESC 'specifies the endpoint address alias as specified in H.323'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

Definition & Use: The endpoint's h323-ID alias as defined in ITU-T H.225. This is one of the dialling attributes defined by H.323. (This field is often incorrectly referred to as 'alias' or 'user name' in many endpoints on the market.)

Number of Values: multi

Indexing Profile: equality

Example Application Using Attribute: white pages, directory of directories, a web page that displays a user's correct configuration information.

Example (LDIF fragment): h323Identityh323-ID: johnsmith
h323Identityh323-ID: conferenceroom201

8.3.3 h323IdentitydialedDigits

OID: 0.0.8.350.1.1.3.1.3
attributetypes: (0.0.8.350.1.1.3.1.3
NAME 'h323IdentitydialedDigits'
DESC 'Specifies the endpoint dialled digits as specified in H.323'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

Definition & Use: The endpoint's H.323 dialedDigits alias as defined in ITU-T H.225. This is one of the dialling attributes defined by H.323. (This field is often incorrectly referred to as 'extension', 'E164' or 'user number' in many endpoints on the market.)

Number of Values: multi

Indexing Profile: equality

Example Application Using Attribute: white pages, directory of directories, a web page that displays a user's correct configuration information.

Example (LDIF fragment): h323IdentitydialedDigits: 2266126

8.3.4 h323Identityemail-ID

OID: 0.0.8.350.1.1.3.1.4
attributetypes: (0.0.8.350.1.1.3.1.4
NAME 'h323Identityemail-ID'
DESC 'Specifies an H.323 entity that can be reached using H.323'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

Definition & Use: The endpoint's H.323 email-ID alias as defined in ITU-T H.225. This is one of the dialing attributes defined by H.323. In some implementations it may be possible to have this field refer to the commOwner's email address in the enterprise directory.

Number of Values: multi

Indexing Profile: equality

Example Application Using Attribute: white pages, directory of directories, a web page that displays a user's correct configuration information.

Example (LDIF fragment): h323Identityemail-ID: user@university.edu

8.3.5 h323IdentityURL-ID

OID: 0.0.8.350.1.1.3.1.5
attributetypes: (0.0.8.350.1.1.3.1.5
NAME 'h323IdentityURL-ID'
DESC 'H.323 specs'
EQUALITY caseExactMatch
SUBSTR caseExactSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

Definition & Use: The endpoint's H.323 URL-ID alias as defined in ITU-T H.323 version 4. This is one of the dialing attributes defined by H.323. The H.323 URL has the general form of user@hostport where either both of the parts (i.e. user and host) or only one of the parts (i.e. user alone or @host alone) is present. The user part corresponds to an H.323 user or service name. The host part is a legal numeric IP address or a fully qualified domain name, thus providing means for address resolution using the DNS infrastructure. Examples include h323:9198437008, h323:dumbledore@gatekeeper.hsww.edu, h323:dumbledore@152.2.2.203, etc. Note that this dialing mechanism is expected to become the preferred addressing scheme for H.323.

Number of Values: multi

Indexing Profile: equality

Example Application Using Attribute: white pages, directory of directories, a web page that displays a user's correct configuration information.

Example (LDIF fragment): h323IdentityURL-ID: h323:dumbledore@gatekeeper.hsww.edu

8.3.6 h323IdentitytransportID

OID: 0.0.8.350.1.1.3.1.6
attributetypes: (0.0.8.350.1.1.3.1.6
NAME 'h323IdentitytransportID'
DESC 'specifies endpoint transport Id as defined in H.323'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

Definition & Use: The endpoint's H.323 transport ID as defined in ITU-T H.225. This is one of the dialing attributes defined by H.323.

Number of Values: multi

Indexing Profile: equality

Example (LDIF fragment): h323IdentitytransportID: 161.58.151.216

8.3.7 h323IdentitypartyNumber

OID: 0.0.8.350.1.1.3.1.7
attributetypes: (0.0.8.350.1.1.3.1.7
NAME 'h323IdentitypartyNumber'
DESC 'endpoint party Number as defined in H.323'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

Definition & Use: The endpoint's H.323 partyNumber alias as defined in ITU-T H.225. This is one of the dialing attributes defined by H.323.

Number of Values: multi

Indexing Profile: equality

Example (LDIF fragment): h323IdentitypartyNumber: 2266126

8.3.8 h323IdentitymobileUIM

OID: 0.0.8.350.1.1.3.1.8
attributetypes: (0.0.8.350.1.1.3.1.8
NAME 'h323IdentitymobileUIM'
DESC 'endpoint mobile UIM as defined in H.323 document'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

Definition & Use: The endpoint's H.323 mobileUIM alias as defined in ITU-T H.225. This is one of the dialing attributes defined by H.323.

Number of Values: multi

Indexing Profile: equality

8.3.9 h323IdentityEndpointType

OID: 0.0.8.350.1.1.3.1.9
attributetypes: (0.0.8.350.1.1.3.1.9
NAME 'h323IdentityEndpointType'
DESC 'The endpoint H.323 type as defined in ITU-T H.323v4.'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

Definition & Use: Describes the type of endpoint as defined in H.323. Values must be one of the following: (1) terminal, (2) mcu, or (3) gateway. This attribute can be used to search the directory for the presence of MCUs, gateways or terminals by searching for the presence of attributes of this type.

Number of Values: multi

Indexing Profile: equality

Example (LDIF fragment): h323IdentityEndpointType:gateway

8.3.10 h323IdentityServiceLevel

OID: 0.0.8.350.1.1.3.1.10
attributetypes: (0.0.8.350.1.1.3.1.10
NAME 'h323IdentityServiceLevel'
DESC 'To define services that a user can belong to.'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

Definition & Use: Describes the class of service a user belongs to. This attribute is outside the scope of H.323 and thus can be populated with class definitions meaningful only to the person who invented those definitions. There should be no expectation of interoperability.

Number of Values: multi

Indexing Profile: equality

Example (LDIF fragment): h323IdentityServiceLevel:deluxe

8.4 h235Identity (H.350.2)

OID: 0.0.8.350.1.1.4.2.1
objectclasses: (0.0.8.350.1.1.4.2.1
NAME 'h235Identity'
DESC 'h235Identity object'
SUP top AUXILIARY
MAY (h235IdentityEndpointID $ h235IdentityPassword $
userCertificate $ cACertificate $ authorityRevocationList $
certificateRevocationList $ crossCertificatePair)
)

Definition & Use: This object class is used to represent H.235 (the security profiles associated with H.323) elements. It is an auxiliary class related to H.350 and implementers should review H.350 in detail. The attributes of h235Identiy include H.235 identity, password and certificate elements. These elements can be downloaded to an endpoint for automatic configuration or accessed by a gatekeeper for call signaling and authentication. The h235Identity object class defines two attributes, h235IdentityEndpointID and h235IdentityPassword, which are needed to implement H.235 Annex D. The remaining attributes that are used, and which are already defined in LDAP, are needed to be able to implement H.235 Annex E. Those attributes are userCertificate, cACertificate, authorityRevocationList, certificateRevocationList, and crossCertificatePair. The definitions and purpose of each of those attributes are defined in IETF RFC2256. In practice, there will always be one and only one h235dentityEndpointID attribute for every endpoint. For applications where the endpoint authenticates against

an LDAP directory, this value may be equal to the commUniqueId value defined in the H.350 document.

8.4.1 h235IdentityEndpointID

OID: 0.0.8.350.1.1.4.1.1
attributetypes: (0.0.8.350.1.1.4.1.1
NAME 'h235IdentityEndpointID'
DESC 'The Sender ID as defined in ITU-H235.'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

Definition & Use: The endpoint's senderID as defined in ITU-H235. This is always identical to endpointID.

Number of Values: multi

Indexing Profile: No recommendation

Example (LDIF fragment): h235IdentityEndpointID: johnsmith

8.4.2 h235IdentityPassword

OID: 0.0.8.350.1.1.4.1.2
attributetypes: (0.0.8.350.1.1.4.1.2
NAME 'h235IdentityPassword'
DESC 'The endpoint password as defined in ITU-H325.'
EQUALITY octetStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.40)

Definition & Use: The endpoint's H.323 password as defined in ITU-T H.235. In practice, there will always be one and only one h235IdentityPassword attribute for every endpoint. If the password is stored in LDAP in encrypted format, then the LDAP encryption algorithm should match the encryption algorithm for the gatekeeper and endpoint, i.e. the gatekeeper and endpoint should support the same encryption format as the LDAP server, even as systems are upgraded over time. This is so the endpoint and gatekeeper may derive the unencrypted password in order to perform H.235 Annex D operations. Since this may not always be possible, the password may be stored in LDAP in an unencrypted fashion. In this case, whenever the password is read by a gatekeeper or endpoint, that communication should be transacted over a secure transport mechanism, e.g. TLS. (See further discussion in Chapter XX on Authentication)

Number of Values: multi

Indexing Profile: equality

Example (LDIF fragment): h235IdentityPassword: 36zxJmCIB18dM0FVAj

8.5 h320Identity (H.350.3)

OID: 0.0.8.350.1.1.5.2.1
objectclasses: (0.0.8.350.1.1.5.2.1
NAME 'h320Identity'
DESC 'h320Identity object'
SUP top AUXILIARY
MAY (h320IdentityCC $ h320IdentityNDC $ h320IdentitySN $
h320IdentityServiceLevel $ h320IdentityExtension)
)

Definition & Use: The h320Identity object class represents H.320 terminals. It is an auxiliary class and is derived from the commObject class derived in H.350. Implementers should review H.350 in detail. The only attribute is described is the GSTN address of the terminal. Note that in this architecture an international public telecommunications number is broken down into its component parts of CC+NDC+SN as defined in E.164. These addresses can be downloaded to an endpoint for automatic configuration or published to white pages to create user dialing directories.

8.5.1 h320IdentityCC

OID: 0.0.8.350.1.1.5.1.1
attributetypes: (0.0.8.350.1.1.5.1.1
NAME 'h320IdentityCC'
DESC 'Country Code'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{3}
)

Definition & Use: Country Code portion of the terminal address as defined in E.164. May also be used for voice numbers.

Number of Values: multi

Indexing Profile: presence, equality, sub

Example Application Using Attribute: A white pages directory that displays a user's ISDN visual telephone address.

Example (LDIF fragment): h320IdentityCC: 1

8.5.2 h320IdentityNDC

OID: 0.0.8.350.1.1.5.1.4
attributetypes: (0.0.8.350.1.1.5.1.4
NAME 'h320IdentityNDC'
DESC 'National Destination Code'
EQUALITY caseIgnoreIA5Match

SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{15}
)

Definition & Use: National Destination Code portion of the terminal address as defined in E.164. May also be used for voice numbers. For example, in the US, the NDC is the area code.

Number of Values: multi

Example Application Using Attribute: A white pages directory that displays a user's ISDN visual telephone address.

Example (LDIF fragment): h320IdentityNDC: 919

Indexing Profile: presence, equality, sub

8.5.3 h320IdentitySN

OID: 0.0.8.350.1.1.5.1.5
attributetypes: (0.0.8.350.1.1.5.1.5
NAME 'h320IdentitySN'
DESC 'Subscriber Number'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26{15}
)

Definition & Use: Subscriber Number portion of the terminal address as defined in E.164. May also be used for voice numbers.

Number of Values: multi

Example Application Using Attribute: A white pages directory that displays a user's ISDN visual telephone address.

Example (LDIF fragment): h320IdentitySN: 1234567

Indexing Profile: presence, equality, sub

8.5.4 h320IdentityExtension

OID: 0.0.8.350.1.1.5.1.3
attributetypes: (0.0.8.350.1.1.5.1.3
NAME 'h320IdentityExtension'
DESC 'Extension of terminal required to dial after initial PSTN address is connected.'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{120}
)

Definition & Use: Specifies an optional extension to be dialled after the PSTN address. May also be used for voice numbers. This attribute can accommodate non-numeric characters, allowing for automatic dialling of extensions. For example, an extension of 1234 that is reachable via Interactive Voice Response, followed by a pound sign, could be represented as ,1234# where the comma indicates the automatic dialler should pause, and the pound sign indicates end of dial string to the IVR. The specific function of digits and characters is not defined here. Note that if the CC+NDC+SN address terminates in a gateway to an IP network, it may be desirable to dial a valid IP address or URL for call completion on the Internet.

Number of Values: multi

Indexing Profile: presence, equality, sub

Example Application Using Attribute: A white pages directory that displays a user's ISDN visual telephone address, including instructions for dialling through an IVR.

Example (LDIF fragment): h320IdentityExtension: 71002
h320IdentityExtension: ,1234#
h320IdentityExtension: h323:user@gatekeeper.foo.com
h320IdentityExtension: 127.0.0.1

8.5.5 h320IdentityServiceLevel

OID: 0.0.8.350.1.1.5.1.2
attributetypes: (0.0.8.350.1.1.5.1.2
NAME 'h320IdentityServiceLevel'
DESC 'To define services that a user can belong to.'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
)

Definition & Use: This describes the type of services a user can belong to. This attribute does not represent a data element found in H.320. Instead, it provides a mechanism for the storage of locally meaningingful authorization information directly in LDAP. For larger applications it may be desirable to ignore this attribute and instead utilize an external authorization server

Number of Values: multi

Indexing Profile: equality

Example Application Using Attribute: Specifying whether certain terminals are authorized to make MCU calls.

Example (LDIF fragment): h320IdentityServiceLevel: premium

8.6 sipIdentity (H.350.4)

OID: 0.0.8.350.1.1.6.2.1
objectclasses: (0.0.8.350.1.1.6.2.1
NAME 'SIPIdentity'
DESC 'SIPIdentity object'
SUP top AUXILIARY
MAY (SIPIdentitySIPURI $ SIPIdentityRegistrarAddress $
SIPIdentityProxyAddress $ SIPIdentityUserName $
SIPIdentityPassword $ SIPIdentityServiceLevel $
userSMIMECertificate)
)

Definition & Use: The SIPIdentity object class represents SIP User Agents (UAs). It is an auxiliary class and is derived from the commObject class and is used to represent SIP User Agents (UAs) on the network and associate those endpoints with users.

8.6.1 SIPIdentitySIPURI

OID: 0.0.8.350.1.1.6.1.1
attributetypes: (0.0.8.350.1.1.6.1.1
NAME 'SIPIdentitySIPURI'
DESC 'Universal Resource Indicator of the SIP UA'
EQUALITY caseExactMatch
SUBSTR caseExactSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

Definition & Use: Uniform Resource Identifier that identifies a communication resource in SIP- usually contains a user name and a host name and is often similar in format to an email address. This URI may institute SIP or SIPS (secure). In the event that SIPS is instituted, the URI must reflect that it is using SIPS as opposed to SIP. See Examples below.

Number of Values: multi

Indexing Profile: equality

Example Application Using Attribute: Online representation of most current listing of a user's SIP(S) UA.

Example (LDIF fragment): SIPIdentitySIPURI: sip:alice@foo.com // SIP example
SIPIdentitySIPURI: sip:alice@152.2.158.212 // SIP example
SIPIdentitySIPURI: sips:bob@birmingham.edu // SIPS example

8.6.2 SIPIdentityRegistrarAddress

OID: 0.0.8.350.1.1.6.1.2
attributetypes: (0.0.8.350.1.1.6.1.2
NAME 'SIPIdentityRegistrarAddress'
DESC 'specifies the location of the registrar'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

Definition & Use: Address for the domain to which the server that handles REGISTER requests and forwarding to the location server for a particular domain belongs. Note that RFC 3261 states that user agents can discover their registrar address by configuration, using the address-of-record, or by multicast. The first scenario, by configuration, is noted as out of scope for RC 3261. This attribute may be used for the first scenario. It can be accomplished manually, (e.g. a web page that displays a user's correct registrar address) or automatically with an H.350.4 aware user agent.

Number of Values: multi

Example Application Using Attribute: white pages, a web page that displays a user's correct configuration information.

Example (LDIF fragment): SIPIdentityRegistrarAddress: 152.2.15.22 //IP address example
SIPIdentityRegistrarAddress: sipregistrar.unc.edu //FQDN example

Indexing Profile: no recommendation

8.6.3 SIPIdentityProxyAddress

OID: 0.0.8.350.1.1.6.1.3
attributetypes: (0.0.8.350.1.1.6.1.3
NAME 'SIPIdentityProxyAddress'
DESC 'Specifies the location of the SIP Proxy'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

Definition & Use: Address that specifies the domain location of SIP proxy within a domain. RFC 3261 defines the role of the SIP proxy. SIP User Agents are not required to use a proxy, but will in many cases.

Number of Values: multi

Example Application Using Attribute: white pages, a web page that displays a user's correct configuration information.

Example (LDIF fragment): SIPIdentityProxyAddress: 172.2.13.234 //IP address example
SIPIdentityProxyAddress: sipproxy.unc.edu //FQDN example

Indexing Profile: no recommendation

8.6.4 SIPIdentityAddress

OID: 0.0.8.350.1.1.6.1.4
attributetypes: (0.0.8.350.1.1.6.1.4
NAME 'SIPIdentityAddress'
DESC 'IP address or FQDN of the UA'
EQUALITY caseIgnoreIA5Match
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

Definition & Use: Specifies the IP address or fully qualified domain name of the UA. This attribute may be useful for applications in which UA to UA communication is direct, not involving a proxy or registrar.

Number of Values: multi

Example Application Using Attribute: A web page that displays a user's proper user agent configuration information.

Example (LDIF fragment): SIPIdentityAddress: 152.2.121.36 // IP address example
SIPIdentityAddress: ipPhone.foo.org // FQDN example

Indexing Profile: equality

8.6.5 SIPIdentityPassword

OID: 0.0.8.350.1.1.6.1.5
attributetypes: (0.0.8.350.1.1.6.1.5
NAME 'SIPIdentityPassword'
DESC 'The user agent SIP password '
EQUALITY octetStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.40)

Definition & Use: The SIP user agent's password, used for the HTTP digest authentication scheme as defined in RFC 2617. Because RFC 2069, which was made obsolete by RFC 2617, was used as the basis for HTTP Digest in RFC 2543, any SIP servers supporting RFC 2617 must ensure backward compatibility with RFC 2069. SIPIdentityUserName, together with SIPIdentityPassword, are reserved for the purpose of use with Digest Access Authentication, and not intended for use with Basic Authentication methods. LDAP provides one method to store user passwords for reference. If passwords are stored in LDAP it makes the LDAP server a particularly valuable target for attack. Implementers are encouraged to exercise caution and implement appropriate security procedures such as encryption, access control, and transport layer security for access to this attribute. (See further discussion in chapter on Authentication)

Number of Values: multi

Indexing Profile: no recommendation

Example (LDIF fragment): SIPIdentityPassword: 36zxJmCIB18dM0FVAj

8.6.6 SIPIdentityUserName

OID: 0.0.8.350.1.1.6.1.6
attributetypes: (0.0.8.350.1.1.6.1.6
NAME 'SIPIdentityUserName'
DESC 'The user agent user name.'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

Definition & Use: The SIP user agent's user name, used for the HTTP digest authentication scheme as defined in RFC 2617. Note that in many cases the user name will be parsed from the user@proxy.domain portion of the SIP URI. In that case it may not be necessary to populate this attribute. Because RFC 2069, which was made obsolete by RFC 2617, was used as the basis for HTTP Digest Authentication in RFC 2543, any SIP servers supporting HTTP Digest Authentication as defined in RFC 2617 must ensure backward compatibility with RFC 2069. This SIPIdentityUserName, together with SIPIdentityPassword, are reserved for the purpose of use with Digest Access Authentication, and not intended for use with Basic Authentication methods.

Number of Values: multi

Indexing Profile: equality

Example (LDIF fragment): SIPIdentityUserName: nelkhour

8.6.7 SIPIdentityServiceLevel

OID: 0.0.8.350.1.1.6.1.7
attributetypes: (0.0.8.350.1.1.6.1.7
NAME 'SIPIdentityServiceLevel'
DESC 'To define services that a user can belong to.'
EQUALITY caseIgnoreIA5Match
SUBSTR caseIgnoreIA5SubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26)

Definition & Use: This describes the level of services a user can belong to. This attribute does not represent a data element found in SIP; SIP itself does not support distinctions in service levels. Instead, this attribute provides a mechanism for the storage of locally meaningful service level

information directly in LDAP. This mapping allows service providers to adapt to an existing LDAP directory without changing the values of the SIPIdentityServiceLevel instances in the directory.

Number of Values: multi

Indexing Profile: equality

Example (LDIF fragment): SIPIdentityServiceLevel: premium

8.7 genericIdentity (H.350.5 Directory Services for Non-Standard Protocols)

OID: 0.0.8.350.1.1.7.2.1
objectclasses: (0.0.8.350.1.1.7.2.1
NAME 'genericIdentity'
DESC 'genericIdentity object'
SUP top AUXILIARY
MAY (genericIdentityProtocolIdentifier $ genericIdentityMessage)
)

Definition & Use: The genericIdentity object class represents other multimedia conferencing information associated with a person or resource. It is an auxiliary class and is related to the commObject class and . implementers should review H.350 in detail. The particular user or resource with which an endpoint is associated via commOwner takes on special importance, as that may represent contact information required for further information in the use of the particular endpoint. If specific attributes such as IP address or URIs are necessary to support this endpoint type, then the standard attributes defining IP address and URI should be used. Keep in mind that in a directory of directories scenario, external searches will only be aware of the genericIdentity attributes and will not know to display IP address or URI. Standardized protocols should not extend and use genericIdentity but should instead create and standardize their own protocol-specific auxiliary classes as new contributions to the H.350 series of recommendations.

8.7.1 genericIdentityProtocolIdentifier

OID: 0.0.8.350.1.1.7.1.1
attributetypes: (0.0.8.350.1.1.7.1.1
NAME 'genericIdentityProtocolIdentifier'
DESC 'name of the non-standard protocol'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

Definition & Use: Text string indicating the name of the non-standard multimedia conferencing services represented by this endpoint that are not H.323, H.320, or SIP; for example: MPEG2, VRVS, Access Grid or other IP Multicast service; Instant Messaging Service

Number of Values: multi

Indexing Profile: equality

Example Application Using Attribute: Search for endpoints that support a specific non-standard protocol. Could be used to locate users of the Access Grid.

Example (LDIF fragment): genericIdentityProtocolIdentifier: 'MPEG2' //MPEG2 endpoint
genericIdentityProtocolIdentifier: 'AccessGrid' //AG user
genericIdentityProtocolIdentifier: 'VRVS' //VRVS user

8.7.2 genericIdentityMessage

OID: 0.0.8.350.1.1.7.1.2
attributetypes: (0.0.8.350.1.1.7.1.2
NAME 'genericIdentityMessage'
DESC 'informative text string'
EQUALITY caseIgnoreMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15)

Definition & Use: Informative text string containing information about multimedia conferencing capabilities of the associated user and/or location of the service. This information may include instructions, other connection information, or pointers to specific documentation.

Number of Values: multi

Indexing Profile: no recommendation

Example (LDIF fragment): genericIdentityMessage: 'see www.foo.com/mpeg2 for connection instructions'
genericIdentityMessage: 'see www.vrvs.org for subscription and bridging instructions'

8.8 LDIF Files

8.8.1 About LDIF Files

All of the LDIF files not included with the Directory Servers or available elsewhere can be found here.

8.8.2 LDIF for commObjectURI

commURI (H.350) is a pointer and method to link people or resources to their multimedia conferencing elements.

LDIF file (for SunOne / iPlanet Directory Servers)
LDIF file (for OpenLDAP Directory Servers)

8.8.3 LDIF for commObject

commObject (H.350) is a generic super class object class from which other protocol specific object classes derive their basic and general functionality.

LDIF file (for SunOne / iPlanet Directory Servers)
LDIF file (for OpenLDAP Directory Servers)

8.8.4 LDIF for h323Identity

H.350.1 is derived from H.350 and defines the h323Identity object class which represents H.323 protocol elements.

LDIF file (for SunOne / iPlanet Directory Servers)
LDIF file (for OpenLDAP Directory Servers)

8.8.5 LDIF for h235Identity

H.350.2 is derived from H.350 and defines the h235Identity object class which represents H.235 protocol elements. This is typically used in conjunction with h323Identity or h320Identity to represent security elements.

LDIF file (for SunOne / iPlanet Directory Servers)
LDIF file (for OpenLDAP Directory Servers)

8.8.6 LDIF for h320Identity

H.350.3 is derived from H.350 and defines the h320Identity object class which represents H.320 protocol elements.

LDIF file (for SunOne / iPlanet Directory Servers)
LDIF file (for OpenLDAP Directory Servers)

8.8.7 LDIF for SIPIdentity

H.350.4 is derived from H.350 and defines the sipIdentity object class which represents SIP protocol elements.

LDIF file (for SunOne / iPlanet Directory Servers)
LDIF file (for OpenLDAP Directory Servers)

8.8.8 LDIF for genericIdentity

LDIF file (for SunOne / iPlanet Directory Servers)
LDIF file (for OpenLDAP Directory Servers)

8.8.9 LDIF for inetOrgperson

LDIF file (for SunOne / iPlanet Directory Servers) - This file is included with the Directory Server

LDIF file (for OpenLDAP Directory Servers) - This file is included with the Directory Server

8.8.10 LDIF for eduPerson

LDIF file (for SunOne / iPlanet Directory Servers) - You can obtain this file from Educause.

LDIF file (for OpenLDAP Directory Servers)

9 Installing and Configuring your Directory Server: Sun ONE (iPlanet) Directory Server

9.1 Installing the Sun ONE (iPlanet) Directory Server

9.1.1 Notes

- The names 'iPlanet' and 'Sun ONE' can be used interchangeably (and are used so by Sun) in most cases. The directory server software began under the name 'iPlanet' and has since moved to 'Sun ONE' under new management. Detailed documents can always be found at Sun One.
- This cookbook and the guide found at Sun ONE are not applicable to the iPlanet Directory Server that is pre-installed on the Solaris[tm] 9 Operating System. Please refer the the System Administration Guide: Naming and Directory Services, Vol. 5 for more information on configuring the pre-installed iPlanet Directory Server.
- The iPlanet server can be run on a 64-bit Solaris[tm] 8 environment, but will only run as a 32-bit process and will be limited to 3.7GB of process memory.
- The iPlanet server will not run on SPARCv8 or earlier chipsets.
- The number of concurrent connections available to the iPlanet Directory Server is limited by the maximum file descriptor table size setting in Solaris[tm]. The variable rlim_fd_max in /etc/system specifies this setting. The value can be changed (or added if it does not already exist) up to 4096 by manually editing the file and rebooting the server. Values above

4096 may cause stability problems and should not be attempted without contacting a Sun Solaris support representative.

9.1.2 Requirements

- Sun Solaris[tm] 9 or 8 Operating Environments (SPARC® Platform Edition)
- At least 2GB for minimal install
- Roughly 4GB or greater for large directories
- At least 256 MB of RAM for minimal install
- Roughly 256 MB to 1 GB or RAM for large directories
- The server must have a static IP Address
- DNS must be properly configured on the server
- You should be familiar with A Recipe for Configuring and Operating LDAP Directories.

9.1.3 Components

- Directory Server
  - This is the actual LDAP directory server.
- Administration Server
  - This is a common front-end to access the all iPlanet servers including the iPlanet Directory server.
- Administration Console
  - This is a common user interface which uses the administration server to communicate with the directory server.

9.1.4 Installation Decisions

There are several decisions one must make prior to installing the Sun ONE Directory Server 5.1.

- Installation Location
- System User
  - The Sun ONE directory server instance will run as this user.
- System Group

  - The Sun ONE directory server instance will run as this group.

- Administrator Identification
  - The administration console requires this login.
- Administrator Password
  - The password for the administrator identification.
- Suffix
  - The suffix is the root of your tree.
- Directory Manager DN
  - The administrative user used for some directory services
- Directory Manager Password
  - The password for the administrative user.
- Administration Domain
  - A part of the configuration directory used to store information about Sun ONE software
- Disable Schema Checking
- Administration Port
  - The port on which to run the administration server.
- Run Administration Server As
  - The user under which to run the administration server.

9.1.5 Installation

(1) Download the latest SunOne Directory Server installation file for Solaris.
(2) Make a directory in which to uncompress the installation file.
(3) The tar file supplied does not automatically expand into a single directory.
(4) Move the file into the installation directory.
(5) Change directories into the installation directory.
(6) Expand the installation file with 'gunzip <installation file>' and then 'tar xvf <ungziped file>'.
(7) If you are not already the root user, please switch to the root user.
(8) Run the './idsktune' utility that was expanded from the installation file. Make any necessary changes issued by idsktune.

(9) Run the setup file with './setup' from the created directory.

(10) Press 'Enter' when asked if you would like to continue with the installation.

> Sun-Netscape Alliance
> iPlanet Server Products Installation/Uninstallation
> ---
> Welcome to the iPlanet Server Products installation program
> This program will install iPlanet Server Products and the
> iPlanet Console on your computer.
> It is recommended that you have "root" privilege to install the
> software.
> Tips for using the installation program:
> - Press "Enter" to choose the default and go to the next screen
> - Type "Control-B" to go back to the previous screen
> - Type "Control-C" to cancel the installation program
> - You can enter multiple items using commas to separate them.
> For example: 1, 2, 3
> Would you like to continue with installation? [Yes]:

(11) If you agree to the terms of the license, type 'Yes' and press 'Enter'.

> Sun-Netscape Alliance
> iPlanet Server Products Installation/Uninstallation
> ---
> BY INSTALLING THIS SOFTWARE YOU ARE CONSENTING TO BE BOUND BY
> AND ARE BECOMING A PARTY TO THE AGREEMENT FOUND IN THE
> LICENSE.TXT FILE. IF YOU DO NOT AGREE TO ALL OF THE TERMS
> OF THIS AGREEMENT, PLEASE DO NOT INSTALL OR USE THIS SOFTWARE.
> Do you agree to the license terms? [No]:

(12) To install the Sun ONE Server, type '1' and press 'Enter'.

> Sun-Netscape Alliance
> iPlanet Server Products Installation/Uninstallation
> ---
> Select the items you would like to install:
> 1. iPlanet Servers
> Installs iPlanet Servers with the integrated iPlanet Console
> onto your computer.
> 2. iPlanet Console
> Installs iPlanet Console
> as a stand-alone Java application on your computer.

To accept the default shown in brackets, press the Enter key.
Select the component you want to install [1]:

(13) Choose the custom installation by typing '3' and pressing 'Enter'.

Sun-Netscape Alliance
iPlanet Server Products Installation/Uninstallation

Choose an installation type:
1. Express installation
Allows you to quickly install the servers using the most common options and pre-defined defaults. Useful for quick evaluation of the products.
2. Typical installation
Allows you to specify common defaults and options.
3. Custom installation
Allows you to specify more advanced options. This is recommended for experienced server administrators only.
To accept the default shown in brackets, press the Enter key.
Choose an installation type [2]:

(14) Type in the location you wish to install the server and press 'Enter' or just press 'Enter' to select the default location. The directory must not exist on a networked drive. The directory must be empty or must not exist. The directory must not be the same directory from which you are installing.

Sun-Netscape Alliance
iPlanet Server Products Installation/Uninstallation

This program will extract the server files and install them into a directory you specify. That directory is called the server root in the product documentation and will contain the server programs, the Administration Server, and the server configuration files.
To accept the default shown in brackets, press the Enter key.
Install location [/usr/iplanet/servers]:

(15) Specify the components you wish to install by typing them in a comma delimited list or just press 'Enter' to install all of the components. It is recommended that you install all of the components.

Sun-Netscape Alliance
iPlanet Server Products Installation/Uninstallation

---

iPlanet Server Products components:
Components with a number in () contain additional subcomponents

which you can select using subsequent screens.
1. Server Core Components (3)
2. iPlanet Directory Suite (2)
3. Administration Services (2)
Specify the components you wish to install [All]:

(16) Specify the server core components you wish to install by typing them in a comma delimited list or just press 'Enter' to install all of the components. It is recommended that you install all of the components.

Sun-Netscape Alliance
iPlanet Server Products Installation/Uninstallation

Server Core Components components:
Components with a number in () contain additional subcomponents
which you can select using subsequent screens.
1. Server Core Components
2. Core Java classes
3. Java Runtime Environment
Specify the components you wish to install [1, 2, 3]:

(17) Specify the Sun ONE directory suite components you wish to install by typing them in a comma delimited list or just press 'Enter' to install all of the components. It is recommended that you install all of the components.

Sun-Netscape Alliance
iPlanet Server Products Installation/Uninstallation
---

iPlanet Directory Suite components:
Components with a number in () contain additional subcomponents
which you can select using subsequent screens.
1. iPlanet Directory Server
2. iPlanet Directory Server Console
Specify the components you wish to install [1, 2]:

(18) Specify the administration services components you wish to install by typing them in a comma delimited list or just press 'Enter' to install all of the components. It is recommended that you install all of the components.

Sun-Netscape Alliance
iPlanet Server Products Installation/Uninstallation

Administration Services components:
Components with a number in () contain additional subcomponents

which you can select using subsequent screens.
1. iPlanet Administration Server
2. Administration Server Console
Specify the components you wish to install [1, 2]:

(19) Enter the fully qualified domain name of your computer or just press 'Enter' to select the default shown in brackets.

Sun-Netscape Alliance
iPlanet Server Products Installation/Uninstallation

Enter the fully qualified domain name of the computer
on which you're installing server software. Using the form
.
Example: eros.airius.com.
To accept the default shown in brackets, press the Enter key.
Computer name [localhost.localdomain]:

(20) Type the name of the user you wish to run the server or just press enter to select the default user 'nobody'.

Sun-Netscape Alliance
iPlanet Server Products Installation/Uninstallation

Choose a Unix user and group to represent the iPlanet server
in the user directory. The iPlanet server will run as this user.
It is recommended that this user should have no privileges
in the computer network system. The Administration Server
will give this group some permissions in the server root
to perform server-specific operations.
If you have not yet created a user and group for the iPlanet
server,create this user and group using your native UNIX
system utilities.
To accept the default shown in brackets, press the Return key.
System User [nobody]:

(21) Type the name of the group you wish to run the server or just press enter to select the default group 'nobody'.

Sun-Netscape Alliance
iPlanet Server Products Installation/Uninstallation

Choose a Unix user and group to represent the iPlanet server
in the user directory. The iPlanet server will run as this user.
It is recommended that this user should have no privileges
in the computer network system. The Administration Server
will give this group some permissions in the server root

to perform server-specific operations.
If you have not yet created a user and group for the iPlanet server,create this user and group using your native UNIX system utilities.
To accept the default shown in brackets, press the Return key.
System User [nobody]:
System Group [nobody]:

(22) Type the name of the administrator you wish to configure for Sun ONE or just press 'Enter' to select the default 'admin'.

Sun-Netscape Alliance
Directory Installation/Uninstallation

In order to reconfigure your installation, the Configuration Directory
Administrator password is required. Here is your current information:
Configuration Directory: ldap://localhost.localdomain:389/o=NetscapeRoot
Configuration Administrator ID: admin
At the prompt, please enter the password for the Configuration Administrator.
iPlanet configuration directory server
administrator ID [admin]:

(23) Type the password you wish to use for the administrator account.

Sun-Netscape Alliance
Directory Installation/Uninstallation

In order to reconfigure your installation, the Configuration Directory
Administrator password is required. Here is your current information:
Configuration Directory: ldap://localhost.localdomain:389/o=NetscapeRoot
Configuration Administrator ID: admin
At the prompt, please enter the password for the Configuration Administrator.
iPlanet configuration directory server
administrator ID [admin]:
Password:

(24) Retype the password for the administrator account.

Sun-Netscape Alliance
Directory Installation/Uninstallation

Please enter the administrator ID for the iPlanet configuration
directory server. This is the ID typically used to log in to the
console. You will also be prompted for the password.
iPlanet configuration directory server
administrator ID [admin]:
Password:
Password (again):

(25) Type in the suffix (i.e. root of your directory tree) for your directory or just press 'Enter' to select the default in brackets.

It is very important to follow DC naming style. Please refer to A Recipe for Configuring and Operating LDAP Directories.

Sun-Netscape Alliance
Directory Installation/Uninstallation

The suffix is the root of your directory tree. You may have more than
one suffix.
Suffix [dc=localdomain]:

(26) Type in the name for the Directory Manager you wish to set up for your directory.

Sun-Netscape Alliance
Directory Installation/Uninstallation

Certain directory server operations require an administrative user.
This user is referred to as the Directory Manager and typically has a
bind Distinguished Name (DN) of cn=Directory Manager. Press Enter to
accept the default value, or enter another DN. In either case, you
will be prompted for the password for this user. The password must
be at least 8 characters long.
Directory Manager DN [cn=Directory Manager]:

(27) Enter a password you wish to use for your Directory Manager.

Sun-Netscape Alliance
Directory Installation/Uninstallation

Certain directory server operations require an administrative user.
This user is referred to as the Directory Manager and typically has a

bind Distinguished Name (DN) of cn=Directory Manager. Press Enter to
accept the default value, or enter another DN. In either case, you will be prompted for the password for this user. The password must
be at least 8 characters long.
Directory Manager DN [cn=Directory Manager]:
Password:

(28) Reenter the password you just typed for your Directory Manager.

Sun-Netscape Alliance
Directory Installation/Uninstallation

Certain directory server operations require an administrative user.
This user is referred to as the Directory Manager and typically has a
bind Distinguished Name (DN) of cn=Directory Manager. Press Enter to
accept the default value, or enter another DN. In either case, you will be prompted for the password for this user. The password must
be at least 8 characters long.
Directory Manager DN [cn=Directory Manager]:
Password:
Password (again):

(29) Type in the Administration Domain for your directory server or just press 'Enter' to select the default in brackets.

Sun-Netscape Alliance
Directory Installation/Uninstallation

The Administration Domain is a part of the configuration directory
server used to store information about iPlanet software. If you are
managing multiple software releases at the same time, or managing
information about multiple domains, you may use the Administration
Domain to keep them separate.
If you are not using administrative domains, press Enter to select the
default. Otherwise, enter some descriptive, unique name for the administration domain, such as the name of the organization responsible

for managing the domain.
Administration Domain [localdomain]:

(30) If you wish to install sample entries into your directory server, type 'Yes' and press 'Enter'. Otherwise, just press 'Enter' to select 'No'.

Sun-Netscape Alliance
Directory Installation/Uninstallation

You may install some sample entries in this directory instance. These
entries will be installed in a separate suffix and will not interfere
with the normal operation of the directory server.
Do you want to install the sample entries? [No]:

(31) If you wish to install any information from and LDIF file at this time, enter the full path to the file and press 'Enter'. If you would like Sun ONE to install some suggested entries, press 'Enter' to select the default 'suggest'. Otherwise, just type 'none' and press 'Enter'. It is preferable to install the suggested entries.

Sun-Netscape Alliance
Directory Installation/Uninstallation

You may wish to populate your new directory instance with some data.
You may already have a file in LDIF format to use or some suggested
entries can be added. If you want to import entries from an LDIF
file, you may type in the full path and filename at the prompt. If
you want the install program to add the suggested entries, type the
word suggest at the prompt. The suggested entries are common
container entries under your specified suffix, such as ou=People and
ou=Groups, which are commonly used to hold the entries for the persons
and groups in your organization. If you do not want to add any of
these entries, type the word none at the prompt.
Type the full path and filename, the word suggest, or the word none
[suggest]:

(32) If you wish to disable schema checking, type 'Yes' and press 'Enter'. Otherwise, just press 'Enter' to select the default 'No'.

Sun-Netscape Alliance
Directory Installation/Uninstallation

If you are going to import an old database immediately after or during
installation, and you think you may have problems with your old schema, you may want to turn off schema checking until after the import. If you choose to do this, schema checking will remain off until you manually turn it back on. iPlanet recommends that you turn
it back on as soon as possible.
Do you want to disable schema checking? [No]:

(33) Enter the adminstration port or press enter to select the default.

Sun-Netscape Alliance
Administration Installation/Uninstallation

The Administration Server is separate from any of your application
servers since it listens to a different port and access to it is restricted.
Pick a port number between 1024 and 65535 to run your Administration
Server on. You should NOT use a port number which you plan to
run an application server on, rather, select a number which you will remember and which will not be used for anything else.
The default in brackets was randomly selected from the available ports on your system. To accept the default, press return.
Administration port [6210]:

(34) Enter in the IP address of your server or press 'Enter' to select the default shown in brackets.

Sun-Netscape Alliance
Administration Installation/Uninstallation

If you want to configure the Administration Server to bind to a specific IP address, enter the address below.
To accept the default shown in brackets, press the Return key.
IP address [ ]:

(35) Enter in the user to run the administration server or press 'Enter' to select the default 'root'.

Sun-Netscape Alliance
Administration Installation/Uninstallation

The Administration Server program runs as a certain user on your
system. This user should be different than the one which your application servers run as. Only the user you select will be able to write to your configuration files. If you run the Administration Server as "root", you will be able to use the Server Administration screen to start and stop your application servers. Run Administration Server as [root]:

(36) The setup script should now start the directory server.

9.2 Configuring an Enterprise Directory with Sun ONE (iPlanet) Directory Server

9.2.1 Notes

These instructions assume that you are working with a clean install of the Sun ONE Directory Server.

9.2.2 Requirements

- A running Sun ONE Directory Server on Solaris
- A running Sun ONE Administration Server on Solaris
- A version of the Sun ONE Administration Console on Solaris
- Knowledge of the following values :
  - The Installation Directory for the Sun ONE Directory Server
  - The Directory Manager Distinguished Name
  - The Directory Manager Password
  - The Domain Name of the server
  - The port number on which the Directory Server is running
- Familiarity with A Recipe for Configuring and Operating LDAP Directories.

9.2.3 Components

- Directory Server
  - This is the actual LDAP directory server.
- Administration Server

  - This is a common front-end to access the all iPlanet servers including the iPlanet Directory server.

- Administration Console
  - This is a common user interface which uses the administration server to communicate with the directory server.

- EduPerson LDIF File
  - This is the file that contains the schema changes necessary to support the EduPerson object.

- commURI LDIF File
  - This is the file that contains the schema changes necessary to support the commURI object.

9.2.4 Legend

Because of the differences in individual needs, certain parameters of the installed Sun ONE Directory Server may be quite different from the typical install. It is for this purpose that the following legend was created.

Parameter	Value	Example
Sun ONE Directory Server Installation Directory	<ldap_dir>	/usr/iplanet/servers
Directory Manager Distinguished Name	<manager_dn>	cn=Directory Manager
Directory Manager Password	<manager_pw>	password
Fully Qualified Domain Name of the Server	<ldap_dns>	enterprise.uab.edu
Sun ONE Directory Server Port Number	<ldap_port>	389
Fully Qualified path to EduPerson LDIF File	<edu_ldif>	/tmp/eduperson.ldif
Fully Qualified path to commURI LDIF File	<comm_ldif>	/tmp/commURI.ldif.txt

9.2.5 Instructions

(1) Download the latest EduPerson LDIF file from Educause and save it locally on your Solaris computer.

(2) Create a copy of the LDIF file and use the new file as your working copy.

(3) Open the LDIF file with a text editor.

(4) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the EduPerson schema to your directory. If you have already added the EduPerson schema to your directory, you can skip this step so that those previously entered values are deleted.

- The code below was taken from the latest EduPerson schema at the time of this writing. If there are additional attributetypes in the EduPerson schema that you have, they will also need to be commented out of the delete function.

delete: attributetypes
attributetypes: (1.3.6.1.4.1.5923.1.1.1.1 NAME 'eduPersonAffiliation')
attributetypes: (1.3.6.1.4.1.5923.1.1.1.2 NAME 'eduPersonNickname')
attributetypes: (1.3.6.1.4.1.5923.1.1.1.3 NAME 'eduPersonOrgDN')
attributetypes: (1.3.6.1.4.1.5923.1.1.1.4 NAME 'eduPersonOrgUnitDN')
attributetypes: (1.3.6.1.4.1.5923.1.1.1.5 NAME 'eduPersonPrimaryAffiliation')
attributetypes: (1.3.6.1.4.1.5923.1.1.1.6 NAME 'eduPersonPrincipalName')
attributetypes: (1.3.6.1.4.1.5923.1.1.1.7 NAME 'eduPersonEntitlement')
attributetypes: (1.3.6.1.4.1.5923.1.1.1.8 NAME 'eduPersonPrimaryOrgUnitDN')
-

(5) The lines mentioned in the previous step should now appear as the following.

 # delete: attributetypes
 # attributetypes: (1.3.6.1.4.1.5923.1.1.1.1 NAME 'eduPersonAffiliation')
 # attributetypes: (1.3.6.1.4.1.5923.1.1.1.2 NAME 'eduPersonNickname')
 # attributetypes: (1.3.6.1.4.1.5923.1.1.1.3 NAME 'eduPersonOrgDN')
 # attributetypes: (1.3.6.1.4.1.5923.1.1.1.4 NAME 'eduPersonOrgUnitDN')
 # attributetypes: (1.3.6.1.4.1.5923.1.1.1.5 NAME 'eduPersonPrimaryAffiliation')
 # attributetypes: (1.3.6.1.4.1.5923.1.1.1.6 NAME 'eduPersonPrincipalName')
 # attributetypes: (1.3.6.1.4.1.5923.1.1.1.7 NAME 'eduPersonEntitlement')
 # attributetypes: (1.3.6.1.4.1.5923.1.1.1.8 NAME 'eduPersonPrimaryOrgUnitDN')
 # -

(6) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the EduPerson schema to your directory. If you have already added the EduPerson schema to your directory, you can skip this step so that those previously entered values are deleted.

delete: objectclasses
objectclasses: (1.3.6.1.4.1.5923.1.1.2
NAME 'eduPerson'
)
-

(7) The lines mentioned in the previous step should now appear as the following.

 # delete: objectclasses
 # objectclasses: (1.3.6.1.4.1.5923.1.1.2
 # NAME 'eduPerson'
 #)
 # -

(8) Change your current working directory to be <ldap_dir>/shared/bin.
(9) Run the following command.

./ldapmodify -D "<manager_dn>" -w <manager_pw> -h <ldap_dns>
\
-p <ldap_port> -f <edu_ldif>

(10) You should see the following feedback from the previous command.

modifying entry cn=schema

(11) The Directory Server should now be updated with the latest EduPerson schema.
(12) Download the latest CommObject LDIF files and save them locally on your Solaris computer.
(13) Create a copy of the commURI.ldif.txt LDIF file and use the new file as your working copy.
(14) Open the LDIF file with a text editor.
(15) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *commObject* schema to your directory. If you have already added the *commObject* schema to your directory, you can skip this step so that those previously entered values are deleted.

- The code below was taken from the latest *commObject* schema at the time of this writing. If there are additional attributetypes in the *commObject* schema that you have, they will also need to be commented out of the delete function.

  delete:attributetypes
  attributetypes: (0.0.8.350.1.1.1.1.1 NAME 'commURI')
  -

(16) The lines mentioned in the previous step should now appear as the following.

 # delete:attributetypes
 # attributetypes: (0.0.8.350.1.1.1.1.1 NAME 'commURI')
 # -

(17) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *commObject* schema to your directory. If you have already added the *commObject* schema to your directory, you can skip this step so that those previously entered values are deleted.
- The code below was taken from the latest *commObject* schema at the time of this writing. If there are additional objectclasses in the *commObject* schema that you have, they will also need to be commented out of the delete function.

  delete: objectclasses
  objectclasses: (0.0.8.350.1.1.1.2.1 NAME 'commURIObject')
  -

(18) The lines mentioned in the previous step should now appear as the following.

 # delete: objectclasses
 # objectclasses: (0.0.8.350.1.1.1.2.1 NAME 'commURIObject')
 # -

(19) Change your current working directory to be <ldap_dir>/shared/bin.
(20) Run the following command.

./ldapmodify -D "<manager_dn>" -w <manager_pw> -h <ldap_dns>
\
-p <ldap_port> -f <comm_ldif>

(21) You should see the following feedback from the previous command.

modifying entry cn=schema

(22) The Directory Server should now be updated with the latest commURIObject schema.

9.3 Configuring an H.350 Directory with Sun ONE (iPlanet) Directory Server

9.3.1 Notes

These instructions assume that you are working with a clean install of the Sun ONE Directory Server.

9.3.2 Requirements

- A running Sun ONE Directory Server on Solaris
- A running Sun ONE Administration Server on Solaris
- A version of the Sun ONE Administration Console on Solaris
- Knowledge of the following values :
  - The Installation Directory for the Sun ONE Directory Server
  - The Directory Manager Distinguished Name
  - The Directory Manager Password
  - The Domain Name of the server
  - The port number on which the Directory Server is running
- Familiarity with A Recipe for Configuring and Operating LDAP Directories.

9.3.3 Components

- Directory Server
  - This is the actual LDAP directory server.
- Administration Server
  - This is a common front-end to access the all iPlanet servers including the iPlanet Directory server.
- Administration Console
  - This is a common user interface which uses the administration server to communicate with the directory server.
- commObject LDIF File
  - This is the file that contains the schema changes necessary to support the *commObject* object.
- h323Identity LDIF File

  - This is the file that contains the schema changes necessary to support the *h323Identity* object.

- h320Identity LDIF File
  - This is the file that contains the schema changes necessary to support the *h320Identity* object.

- h235Identity LDIF File
  - This is the file that contains the schema changes necessary to support the *h235Identity* object.

- sipIdentity LDIF File
  - This is the file that contains the schema changes necessary to support the *sipIdentity* object.

- genericIdentity LDIF File
  - This is the file that contains the schema changes necessary to support the *genericIdentity* object.

9.3.4 Legend

Because of the differences in individual needs, certain parameters of the installed Sun ONE Directory Server may be quite different from the typical install. It is for this purpose that the following legend was created.

Parameter	Value	Example
Sun ONE Directory Server Installation Directory	<ldap_dir>	/usr/iplanet/servers
Directory Manager Distinguished Name	<manager_dn>	cn=Directory Manager
Directory Manager Password	<manager_pw>	password
Fully Qualified Domain Name of the Server	<ldap_dns>	enterprise.uab.edu
Sun ONE Directory Server Port Number	<ldap_port>	389
Fully Qualified path to *commObject* LDIF File	<comm_ldif>	/tmp/commObject.ldif.txt
Fully Qualified path to *h323Identity* LDIF File	<h323_ldif>	/tmp/h323Identity.ldif.txt
Fully Qualified path to h320Identity LDIF File	<h320_ldif>	/tmp/h323Identity.ldif.txt
Fully Qualified path to h235Identity LDIF File	<h235_ldif>	/tmp/h235Identity.ldif.txt
Fully Qualified path to sipIdentity LDIF File	<sip_ldif>	/tmp/sipIdentity.ldif.txt
Fully Qualified path to genericIdentity LDIF File	<gen_ldif>	/tmp/genericIdentity.ldif.txt

9.3.5 Instructions

(1) Download the latest CommObject LDIF files and save them locally on your Solaris computer.

(2) Create a copy of the commObject.ldif.txt LDIF file and use the new file

as your working copy.

(3) Open the LDIF file with a text editor.

(4) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *commObject* schema to your directory. If you have already added the *commObject* schema to your directory, you can skip this step so that those previously entered values are deleted.
- The code below was taken from the latest *commObject* schema at the time of this writing. If there are additional attributetypes in the *commObject* schema that you have, they will also need to be commented out of the delete function.

delete:attributetypes
attributetypes: (0.0.8.350.1.1.2.1.1 NAME 'commUniqueId')
attributetypes: (0.0.8.350.1.1.2.1.2 NAME 'commOwner')
attributetypes: (0.0.8.350.1.1.2.1.3 NAME 'commPrivate')
-

(5) The lines mentioned in the previous step should now appear as the following.

 # delete:attributetypes
 # attributetypes: (0.0.8.350.1.1.2.1.1 NAME 'commUniqueId')
 # attributetypes: (0.0.8.350.1.1.2.1.2 NAME 'commOwner')
 # attributetypes: (0.0.8.350.1.1.2.1.3 NAME 'commPrivate')
 # -

(6) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *commObject* schema to your directory. If you have already added the *commObject* schema to your directory, you can skip this step so that those previously entered values are deleted.
- The code below was taken from the latest *commObject* schema at the time of this writing. If there are additional objectclasses in the *commObject* schema that you have, they will also need to be commented out of the delete function.

delete: objectclasses
objectclasses: (0.0.8.350.1.1.2.2.1 NAME 'commObject')
-

(7) The lines mentioned in the previous step should now appear as the following.

 # delete: objectclasses
 # objectclasses: (0.0.8.350.1.1.2.2.1 NAME 'commObject')
 # -

(8) Change your current working directory to be <ldap_dir>/shared/bin.
(9) Run the following command.

./ldapmodify -D "<manager_dn>" -w <manager_pw> -h <ldap_dns>
**
-p <ldap_port> -f <comm_ldif>

(10) You should see the following feedback from the previous command.

modifying entry cn=schema

(11) The Directory Server should now be updated with the latest *commObject* schema.
(12) Create a copy of the h323Identity.ldif.txt LDIF file and use the new file as your working copy.
(13) Open the LDIF file with a text editor.
(14) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *h323Identity* schema to your directory. If you have already added the *h323Identity* schema to your directory, you can skip this step so that those previously entered values are deleted.
- The code below was taken from the latest *h323Identity* schema at the time of this writing. If there are additional attributetypes in the *h323Identity* schema that you have, they will also need to be commented out of the delete function.

delete:attributetypes
attributetypes: (0.0.8.350.1.1.3.1.1 NAME 'h323IdentityGKDomain'
)
attributetypes: (0.0.8.350.1.1.3.1.2 NAME 'h323Identityh323-ID'
)
attributetypes: (0.0.8.350.1.1.3.1.3 NAME 'h323IdentitydialedDigits'
)
attributetypes: (0.0.8.350.1.1.3.1.4 NAME 'h323Identityemail-ID')
attributetypes: (0.0.8.350.1.1.3.1.5 NAME 'h323IdentityURL-ID'
)
attributetypes: (0.0.8.350.1.1.3.1.6 NAME 'h323IdentitytransportID'
)
attributetypes: (0.0.8.350.1.1.3.1.7 NAME 'h323IdentitypartyNumber'
)

attributetypes: (0.0.8.350.1.1.3.1.8 NAME 'h323IdentitymobileUIM'
)
attributetypes: (0.0.8.350.1.1.3.1.9 NAME 'h323IdentityEndpointType'
)
attributetypes: (0.0.8.350.1.1.3.1.10 NAME 'h323IdentityServiceLevel'
)
-

(15) The lines mentioned in the previous step should now appear as the following.

 # delete:attributetypes
 # attributetypes: (0.0.8.350.1.1.3.1.1 NAME 'h323IdentityGKDomain'
)
 # attributetypes: (0.0.8.350.1.1.3.1.2 NAME 'h323Identityh323-
ID')
 # attributetypes: (0.0.8.350.1.1.3.1.3 NAME 'h323IdentitydialedDigits'
)
 # attributetypes: (0.0.8.350.1.1.3.1.4 NAME 'h323Identityemail-
ID')
 # attributetypes: (0.0.8.350.1.1.3.1.5 NAME 'h323IdentityURL-
ID')
 # attributetypes: (0.0.8.350.1.1.3.1.6 NAME 'h323IdentitytransportID'
)
 # attributetypes: (0.0.8.350.1.1.3.1.7 NAME 'h323IdentitypartyNumber'
)
 # attributetypes: (0.0.8.350.1.1.3.1.8 NAME 'h323IdentitymobileUIM'
)
 # attributetypes: (0.0.8.350.1.1.3.1.9 NAME 'h323IdentityEndpointType'
)
 # attributetypes: (0.0.8.350.1.1.3.1.10 NAME 'h323IdentityServiceLevel'
)
 # -

(16) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *h323Identity* schema to your directory. If you have already added the *h323Identity* schema to your directory, you can skip this step so that those previously entered values are deleted.
- The code below was taken from the latest *h323Identity* schema at the time of this writing. If there are additional objectclasses in the *h323Identity* schema that you have, they will also need to be commented out of the delete function.

delete: objectclasses
objectclasses: (0.0.8.350.1.1.3.2.1 NAME 'h323Identity')
-

(17) The lines mentioned in the previous step should now appear as the following.

 # delete: objectclasses
 # objectclasses: (0.0.8.350.1.1.3.2.1 NAME 'h323Identity')
 # -

(18) Change your current working directory to be <ldap_dir>/shared/bin.
(19) Run the following command.

./ldapmodify -D "<manager_dn>" -w <manager_pw> -h <ldap_dns>
**
-p <ldap_port> -f <h323_ldif>

(20) You should see the following feedback from the previous command.

modifying entry cn=schema

(21) The Directory Server should now be updated with the latest *h323Identity* schema.
(22) Create a copy of the h320Identity.ldif.txt LDIF file and use the new file as your working copy.
(23) Open the LDIF file with a text editor.
(24) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *h320Identity* schema to your directory. If you have already added the *h320Identity* schema to your directory, you can skip this step so that those previously entered values are deleted.
- The code below was taken from the latest *h320Identity* schema at the time of this writing. If there are additional attributetypes in the *h320Identity* schema that you have, they will also need to be commented out of the delete function.

delete:attributetypes
attributetypes: (0.0.8.350.1.1.5.1.1 NAME 'h320IdentityAddress'
)
attributetypes: (0.0.8.350.1.1.5.1.2 NAME 'h320IdentityServiceLevel'
)
-

(25) The lines mentioned in the previous step should now appear as the following.

 # delete:attributetypes
 # attributetypes: (0.0.8.350.1.1.5.1.1 NAME 'h320IdentityAddress'
)
 # attributetypes: (0.0.8.350.1.1.5.1.2 NAME 'h320IdentityServiceLevel'
)
 # -

(26) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *h320Identity* schema to your directory. If you have already added the *h320Identity* schema to your directory, you can skip this step so that those previously entered values are deleted.
- The code below was taken from the latest *h320Identity* schema at the time of this writing. If there are additional objectclasses in the *h320Identity* schema that you have, they will also need to be commented out of the delete function.

delete: objectclasses
objectclasses: (0.0.8.350.1.1.5.2.1 NAME 'h320Identity')
-

(27) The lines mentioned in the previous step should now appear as the following.

 # delete: objectclasses
 # objectclasses: (0.0.8.350.1.1.5.2.1 NAME 'h320Identity')
 # -

(28) Change your current working directory to be <ldap_dir>/shared/bin.
(29) Run the following command.

./ldapmodify -D "<manager_dn>" -w <manager_pw> -h <ldap_dns>
**
-p <ldap_port> -f <h320_ldif>

(30) You should see the following feedback from the previous command.

modifying entry cn=schema

(31) The Directory Server should now be updated with the latest *h320Identity* schema.
(32) Create a copy of the h235Identity.ldif.txt LDIF file and use the new file as your working copy.
(33) Open the LDIF file with a text editor.
(34) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *h235Identity* schema to your directory. If you have already added the *h235Identity* schema to your directory, you can skip this step so that those previously entered values are deleted.
- The code below was taken from the latest *h235Identity* schema at the time of this writing. If there are additional attributetypes in the *h235Identity* schema that you have, they will also need to be commented out of the delete function.

delete:attributetypes
attributetypes: (0.0.8.350.1.1.4.1.1 NAME 'h235IdentitySenderID'
)
attributetypes: (0.0.8.350.1.1.4.1.2 NAME 'h235IdentityPassword'
)
-

(35) The lines mentioned in the previous step should now appear as the following.

 # delete:attributetypes
 # attributetypes: (0.0.8.350.1.1.4.1.1 NAME 'h235IdentitySenderID'
)
 # attributetypes: (0.0.8.350.1.1.4.1.2 NAME 'h235IdentityPassword'
)
 # -

(36) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *h235Identity* schema to your directory. If you have already added the *h235Identity* schema to your directory, you can skip this step so that those previously entered values are deleted.
- The code below was taken from the latest *h235Identity* schema at the time of this writing. If there are additional objectclasses in the *h235Identity* schema that you have, they will also need to be commented out of the delete function.

delete: objectclasses
objectclasses: (0.0.8.350.1.1.4.2.1 NAME 'h235Identity')
-

(37) The lines mentioned in the previous step should now appear as the following.

 # delete: objectclasses
 # objectclasses: (0.0.8.350.1.1.4.2.1 NAME 'h235Identity')
 # -

(38) Change your current working directory to be <ldap_dir>/shared/bin.
(39) Run the following command.

./ldapmodify -D "<manager_dn>" -w <manager_pw> -h <ldap_dns>
\
-p <ldap_port> -f <h235_ldif>

(40) You should see the following feedback from the previous command.

modifying entry cn=schema

(41) The Directory Server should now be updated with the latest *h235Identity* schema.
(42) Create a copy of the sipIdentity.ldif.txt LDIF file and use the new file as your working copy.
(43) Open the LDIF file with a text editor.
(44) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *sipIdentity* schema to your directory. If you have already added the *sipIdentity* schema to your directory, you can skip this step so that those previously entered values are deleted.
- The code below was taken from the latest *sipIdentity* schema at the time of this writing. If there are additional attributetypes in the *sipIdentity* schema that you have, they will also need to be commented out of the delete function.

delete:attributetypes
attributetypes: (0.0.8.350.1.1.6.1.1 NAME 'SIPIdentitySIPURI'
)
attributetypes: (0.0.8.350.1.1.6.1.2 NAME 'SIPIdentityRegistrar-Domain')
attributetypes: (0.0.8.350.1.1.6.1.3 NAME 'SIPIdentityProxyDomain')
attributetypes: (0.0.8.350.1.1.6.1.4 NAME 'SIPIdentityIPAddress'
)
attributetypes: (0.0.8.350.1.1.6.1.5 NAME 'SIPIdentityPassword'
)
attributetypes: (0.0.8.350.1.1.6.1.6 NAME 'SIPIdentityUserName'
)
attributetypes: (0.0.8.350.1.1.6.1.7 NAME 'SIPIdentityServiceLevel'
)
-

(45) The lines mentioned in the previous step should now appear as the following.

 # delete:attributetypes
 # attributetypes: (0.0.8.350.1.1.6.1.1 NAME 'SIPIdentitySIPURI')
 # attributetypes: (0.0.8.350.1.1.6.1.2 NAME 'SIPIdentityRegistrarDomain')
 # attributetypes: (0.0.8.350.1.1.6.1.3 NAME 'SIPIdentityProxyDomain')
 # attributetypes: (0.0.8.350.1.1.6.1.4 NAME 'SIPIdentityIPAddress')
 # attributetypes: (0.0.8.350.1.1.6.1.5 NAME 'SIPIdentityPassword')
 # attributetypes: (0.0.8.350.1.1.6.1.6 NAME 'SIPIdentityUserName')
 # attributetypes: (0.0.8.350.1.1.6.1.7 NAME 'SIPIdentityServiceLevel')
 # -

(46) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *sipIdentity* schema to your directory. If you have already added the *sipIdentity* schema to your directory, you can skip this step so that those previously entered values are deleted.
- The code below was taken from the latest *sipIdentity* schema at the time of this writing. If there are additional objectclasses in the *sipIdentity* schema that you have, they will also need to be commented out of the delete function.

delete: objectclasses
objectclasses: (0.0.8.350.1.1.6.2.1 NAME 'SIPIdentity')
-

(47) The lines mentioned in the previous step should now appear as the following.

 # delete: objectclasses
 # objectclasses: (0.0.8.350.1.1.6.2.1 NAME 'SIPIdentity')
 # -

(48) Change your current working directory to be <ldap_dir>/shared/bin.
(49) Run the following command.

./ldapmodify -D "<manager_dn>" -w <manager_pw> -h <ldap_dns>
**
-p <ldap_port> -f <sip_ldif>

(50) You should see the following feedback from the previous command.

modifying entry cn=schema

(51) The Directory Server should now be updated with the latest *sipIdentity* schema.

(52) Create a copy of the genericIdentity.ldif.txt LDIF file and use the new file as your working copy.

(53) Open the LDIF file with a text editor.

(54) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *sipIdentity* schema to your directory. If you have already added the *sipIdentity* schema to your directory, you can skip this step so that those previously entered values are deleted.
- The code below was taken from the latest *sipIdentity* schema at the time of this writing. If there are additional attributetypes in the *sipIdentity* schema that you have, they will also need to be commented out of the delete function.

delete: attributetypes
attributetypes:(0.0.8.350.1.1.7.1.1 NAME 'genericIdentityProtocolIdentifier')
attributetypes: (0.0.8.350.1.1.7.1.2 NAME 'genericIdentityMessage')
-

(55) The lines mentioned in the previous step should now appear as the following.

 # delete: attributetypes
 # attributetypes:(0.0.8.350.1.1.7.1.1 NAME 'genericIdentityProtocolIdentifier')
 # attributetypes: (0.0.8.350.1.1.7.1.2 NAME 'genericIdentityMessage')
 # -

(56) Comment out the following lines in the file by placing the "#" character in front of the lines.

- The delete function (see below) needs to be commented out if you have not added the *sipIdentity* schema to your directory. If you have already added the *sipIdentity* schema to your directory, you can skip this step so that those previously entered values are deleted.

- The code below was taken from the latest *sipIdentity* schema at the time of this writing. If there are additional objectclasses in the *sipIdentity* schema that you have, they will also need to be commented out of the delete function.

  delete: objectclasses
  objectclasses: (0.0.8.350.1.1.7.2.1 NAME 'genericIdentity')
  -

(57) The lines mentioned in the previous step should now appear as the following.

 # delete: objectclasses
 # objectclasses: (0.0.8.350.1.1.7.2.1 NAME 'genericIdentity')
 # -

(58) Change your current working directory to be <ldap_dir>/shared/bin.
(59) Run the following command.

./ldapmodify -D "<manager_dn>" -w <manager_pw> -h <ldap_dns>
**
-p <ldap_port> -f <gen_ldif>

(60) You should see the following feedback from the previous command.
(61) The Directory Server should now be updated with the latest *genericIdentity* schema.

9.4 Populating an Enterprise Directory with Sun ONE (iPlanet) Directory Server

9.4.1 Notes

None.

9.4.2 Requirements

- A running Sun ONE Directory Server on Solaris
- A running Sun ONE Administration Server on Solaris
- A version of the Sun ONE Administration Console on Solaris
- You should know the values for the following :
  - The Installation Directory for the Sun ONE Directory Server
  - The Directory Manager Distinguished Name
  - The Directory Manager Password
  - The Domain Name of the server
  - The port number on which the Directory Server is running
- You should be familiar with A Recipe for Configuring and Operating LDAP Directories.

9.4.3 Components

- Directory Server
  - This is the actual LDAP directory server.
- Administration Server
  - This is a common front-end to access the all iPlanet servers including the iPlanet Directory server.
- Administration Console
  - This is a common user interface which uses the administration server to communicate with the directory server.

9.4.4 Legend

Because of the differences in individual needs, certain parameters of the installed Sun ONE Directory Server may be quite different from the typical install. It is for this purpose that the following legend was created.

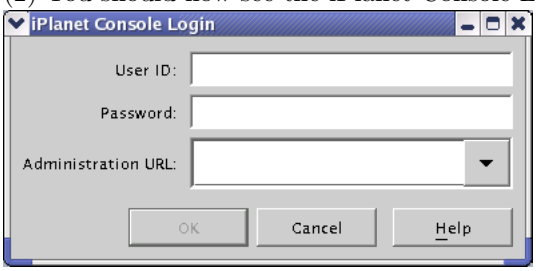
Parameter	Value
Sun ONE Directory Server Installation Directory	<ldap_dir>
Directory Manager Distinguished Name	<manager_dn>
Directory Manager Password	<manager_pw>
Fully Qualified Domain Name of the Server	<ldap_dns>
Sun ONE Directory Server Port Number	<ldap_port>

9.4.5 Instructions

(1) Run the following command from your Sun ONE Directory Server Installation Directory (<ldap_dir>).

./startconsole

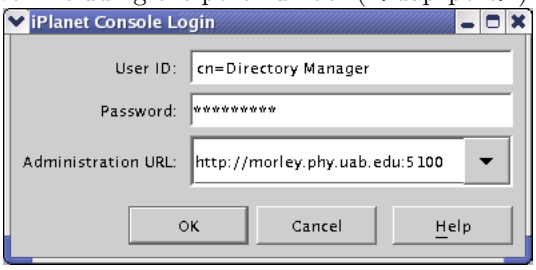
(2) You should now see the iPlanet Console Login window.



(3) In the User ID field, enter the Directory Manager Distinguished Name (<manager_dn>).

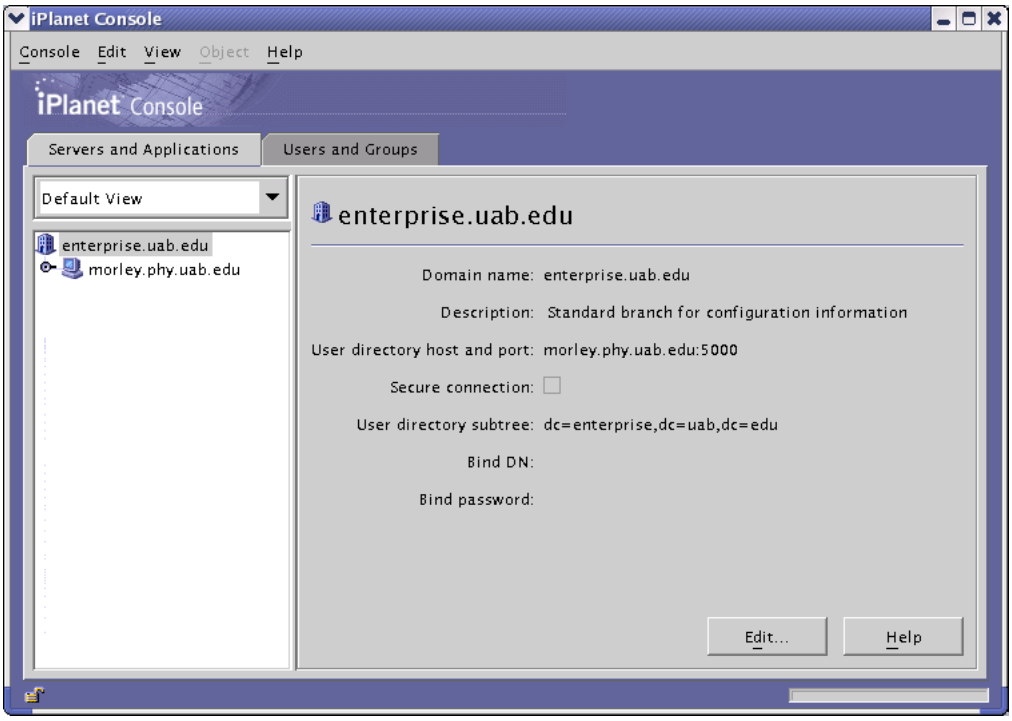
(4) In the Password field, enter the Directory Manager Password (<manager_pw>).

(5) In the Administration URL field, enter in the URL for the directory server including the port number (<ldap_port>).



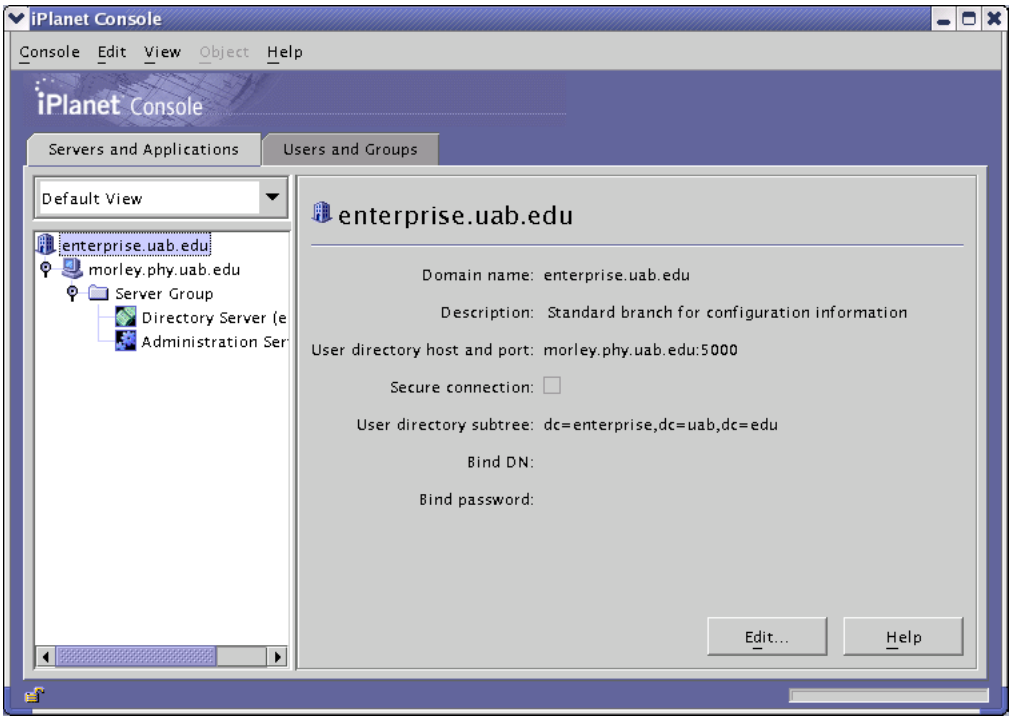
(6) Click OK.

(7) You should now see the iPlanet Console.

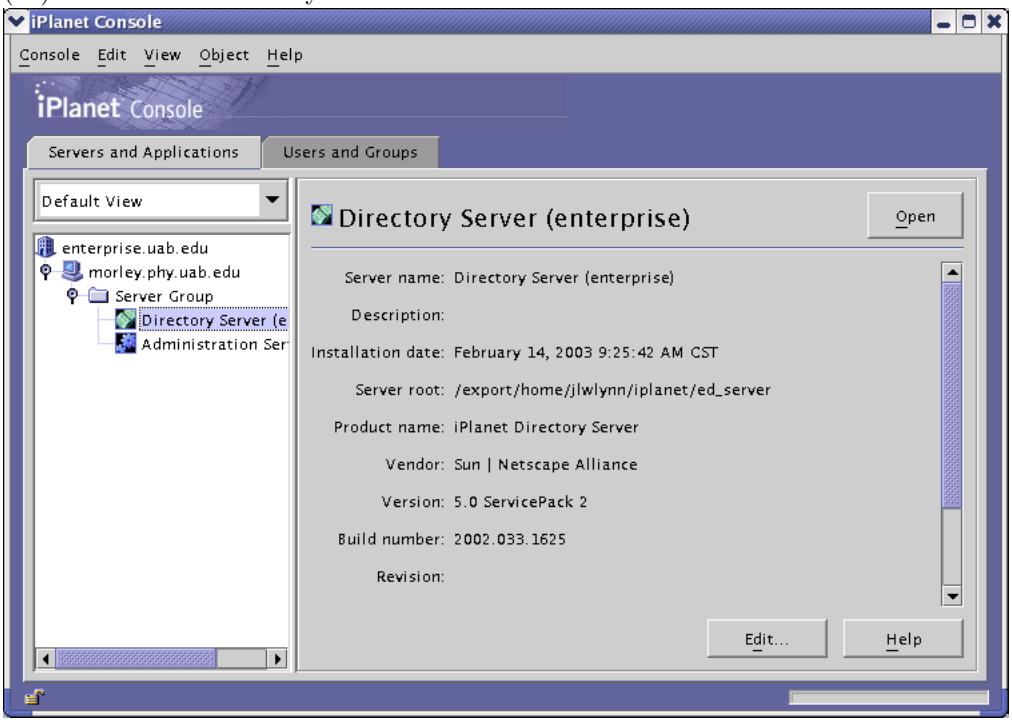


(8) On the left-hand side of the iPlanet Console Desktop, you should see a Java tree structure containing the name of your directory server and directly beneath it is the distinguished name of your server (<ldap_dns>).

(9) Click on the file handles in the tree to open them until you reach the Directory Server node.



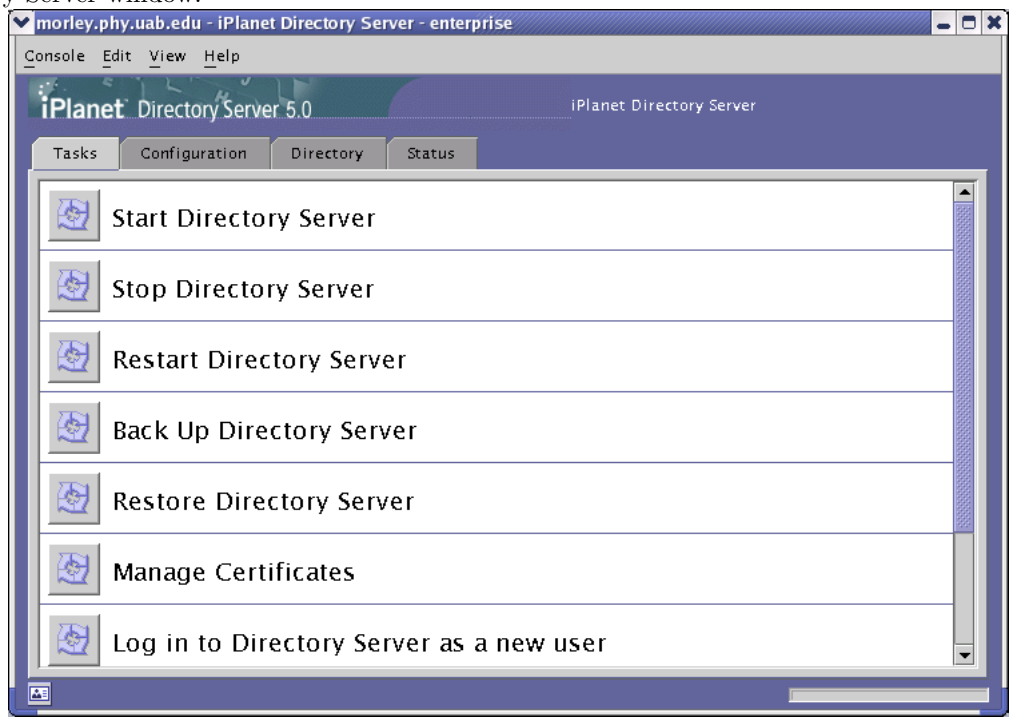
(10) Click on the Directory Server node.



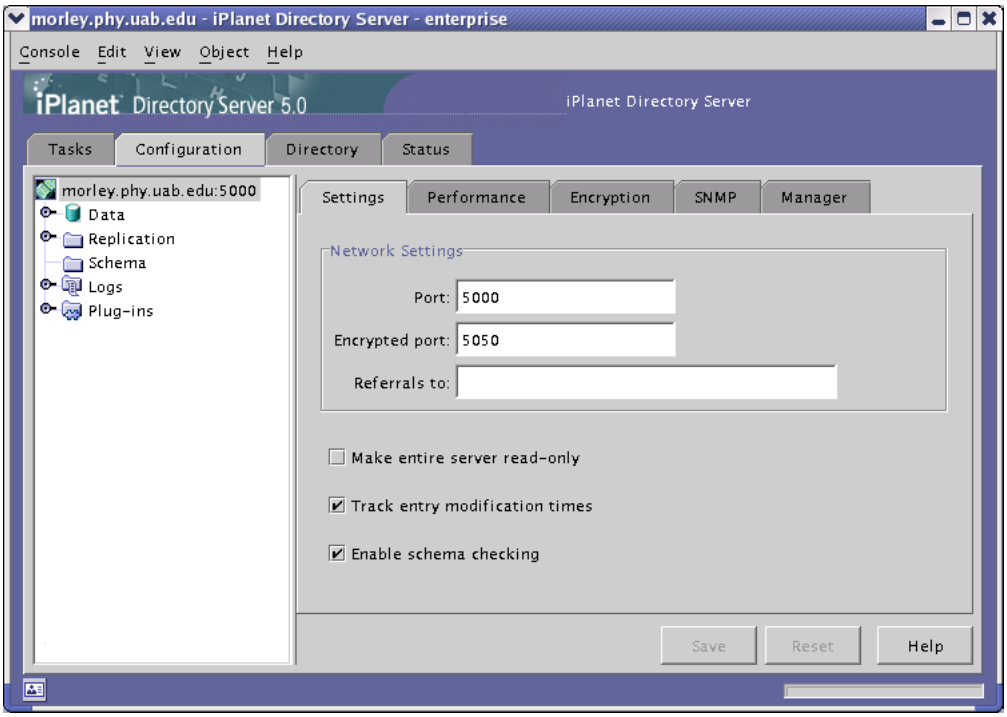
(11) You should now see some basic property information about your directory server on the right-hand side of the iPlanet Console Desktop.

(12) Click on the Open button contained in the property information..

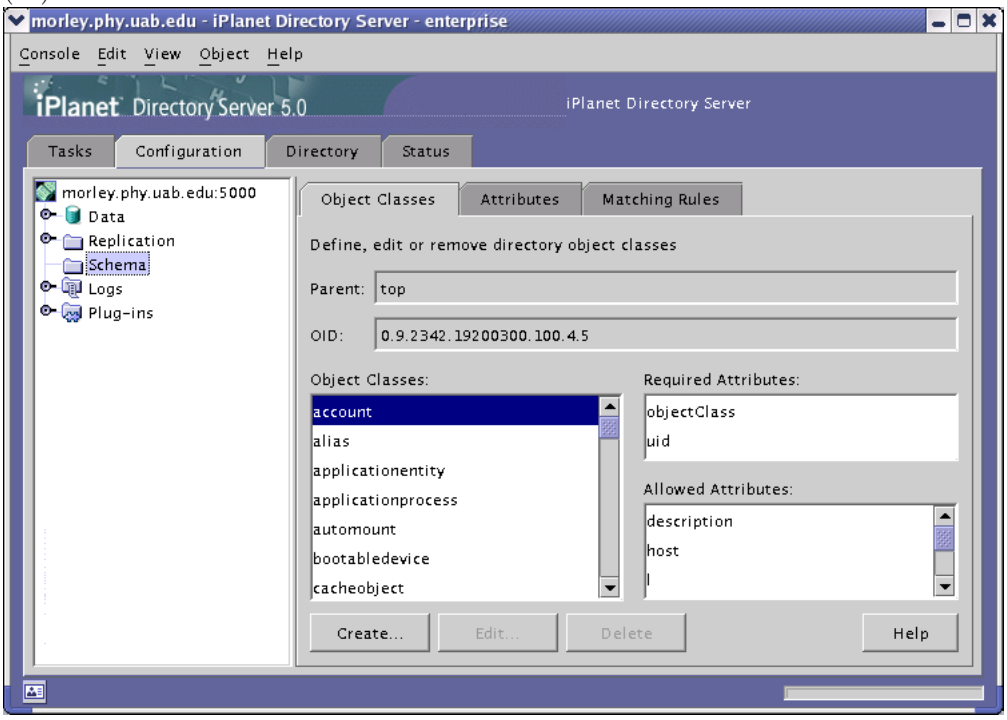
(13) You should now see your directory server open in a new iPlanet Directory Server window.



(14) Click on the Configuration tab in the iPlanet Directory Server window.

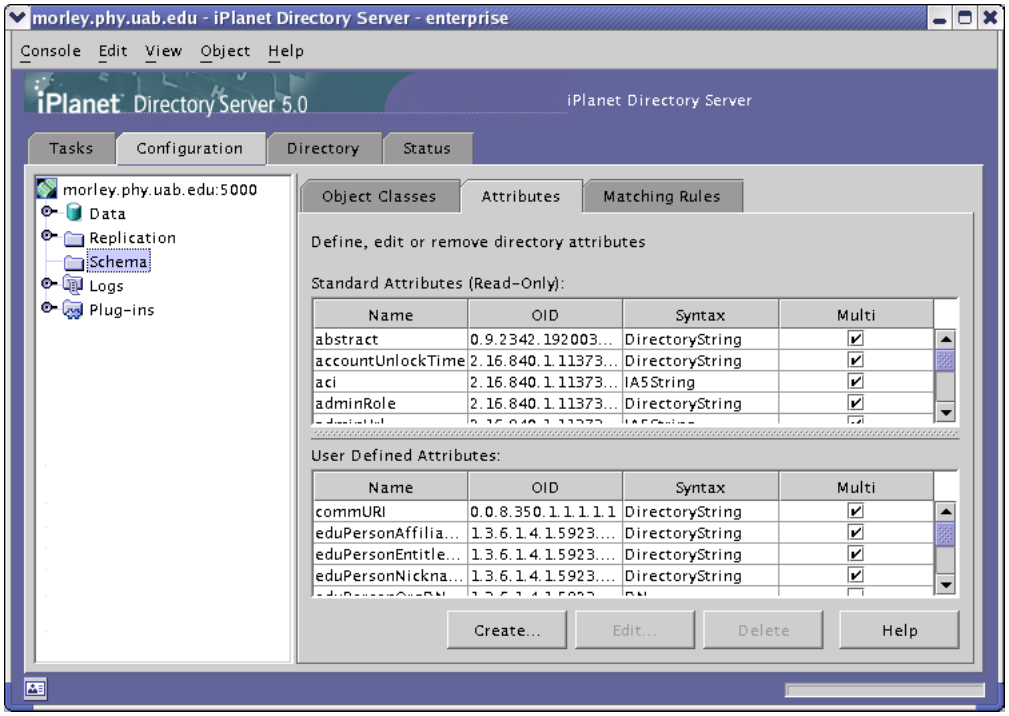


(15) Click on the Schema node in the Java tree structure.



(16) Click on the Attributes tab on the right-hand side of the iPlanet Directory Server Desktop.

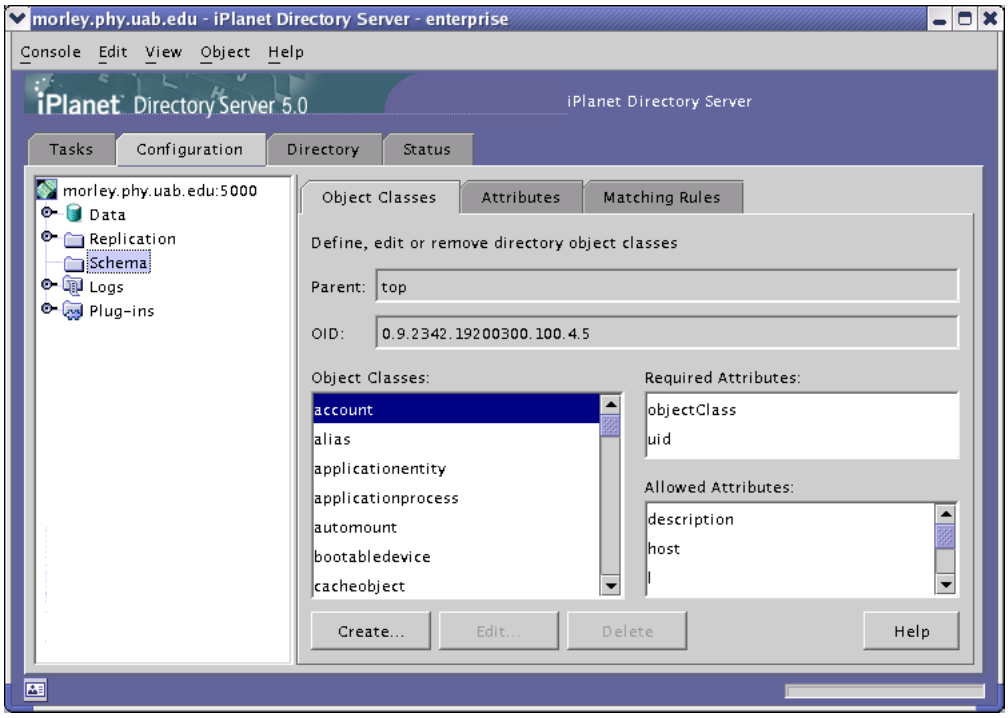
- Using the iPlanet Directory Server to create new objects, you must create the attributes, create the object, then add the attributes to the objects.



(17) Use the Create... button to add any additional attributes that your institution may need.

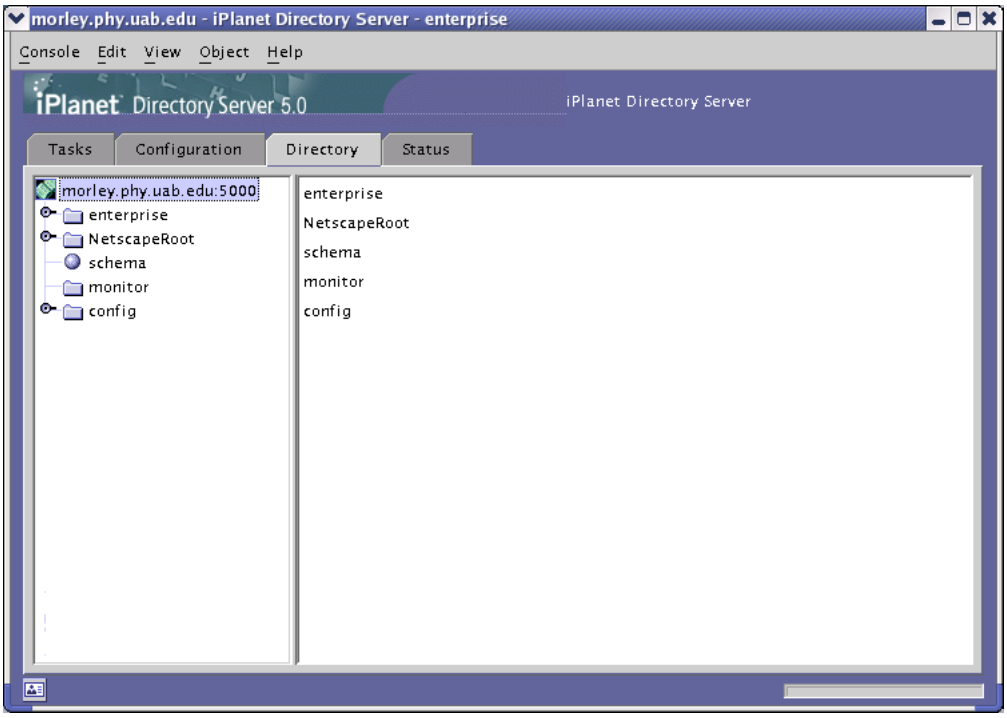
(18) Click on the Object Classes tab on the right-hand side of the iPlanet Directory Server Desktop.

- Using the iPlanet Directory Server to create new objects, you must create the attributes, create the object, then add the attributes to the objects.

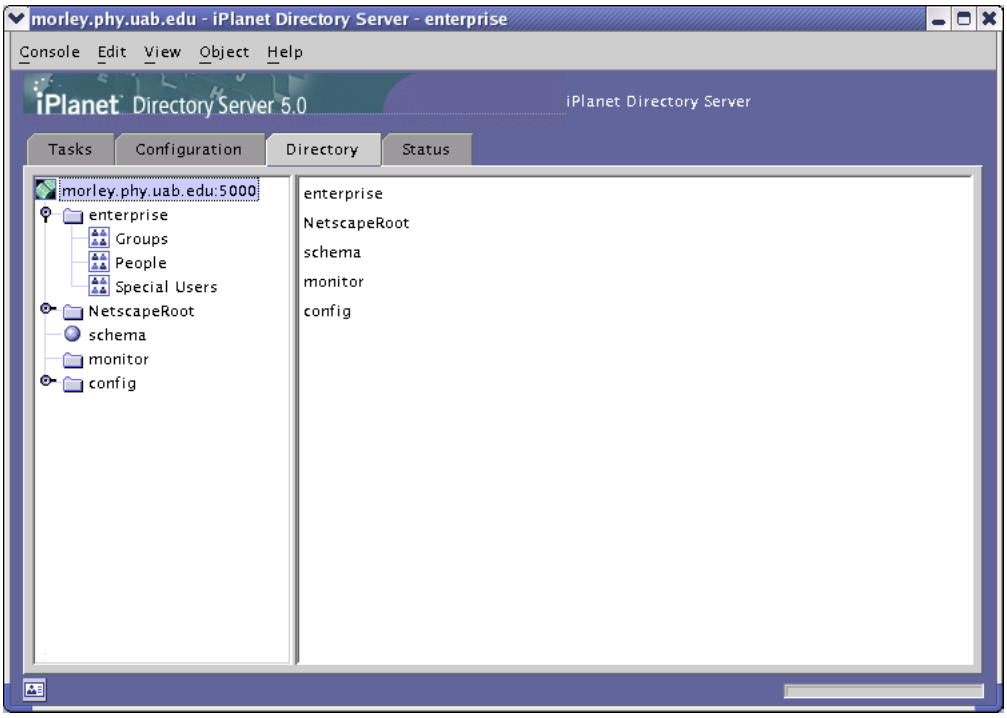


(19) Use the Create... button to add any additional object classes that your institution may need.

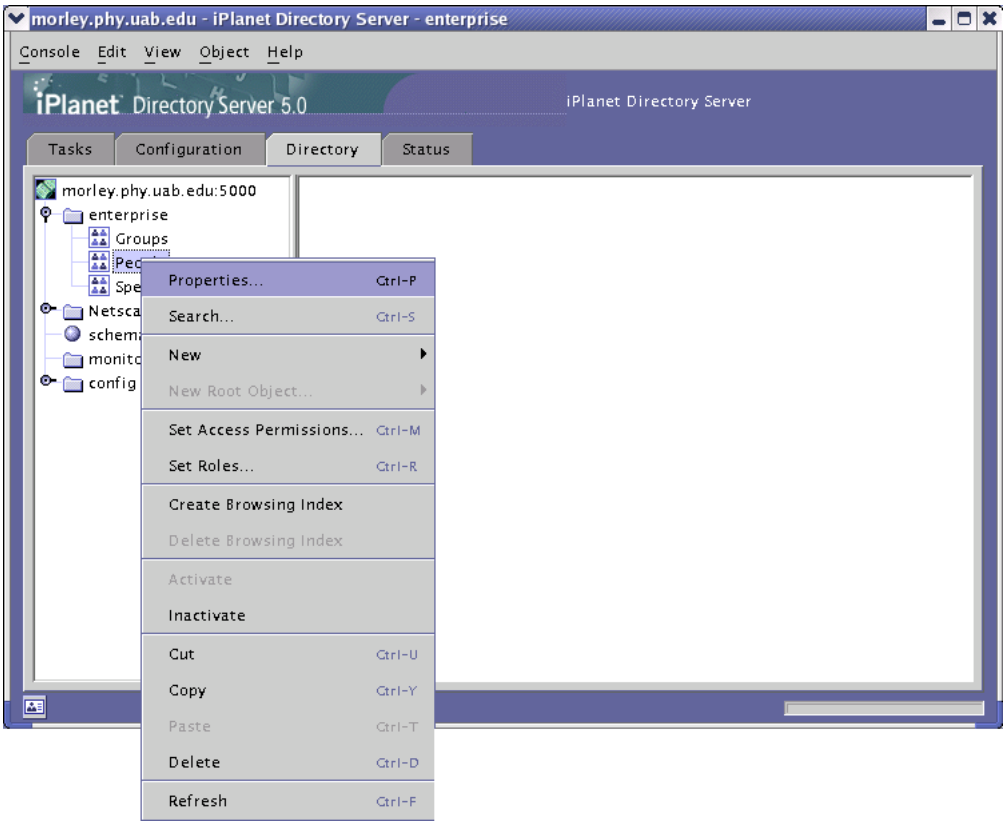
(20) Click on the Directory tab at the top of the iPlanet Directory Server Desktop.



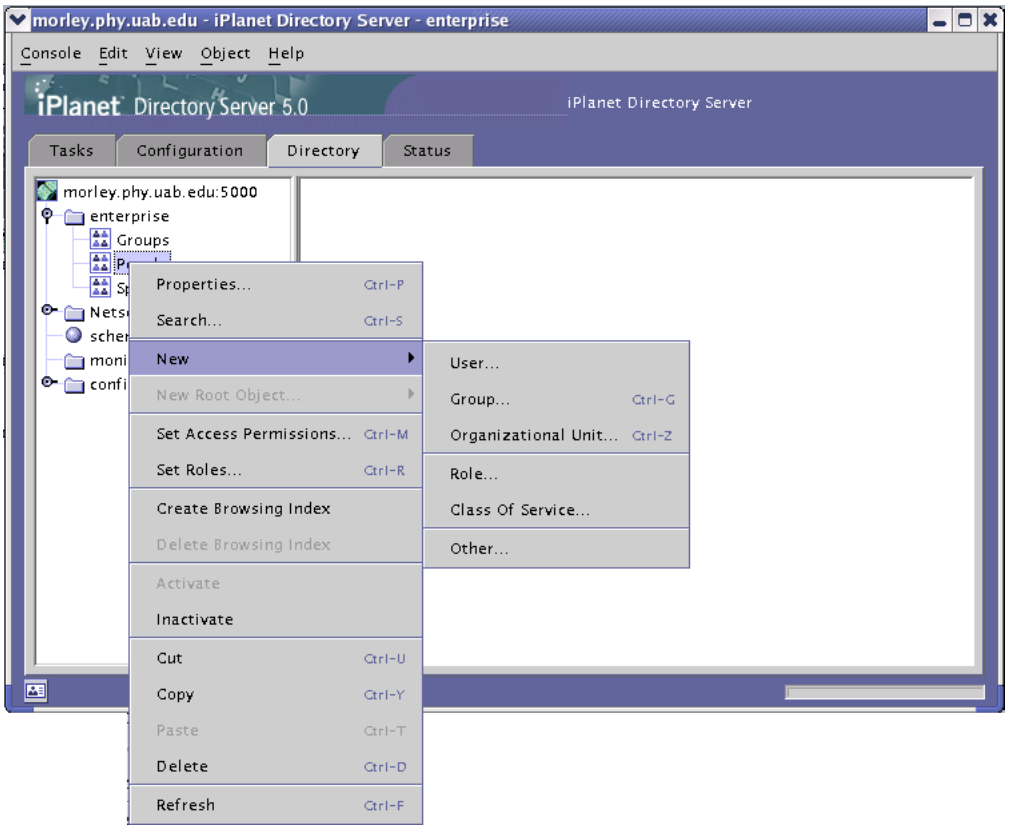
(21) Open your directory data to the Organizational Unit People by navigating to it in the Java tree structure.



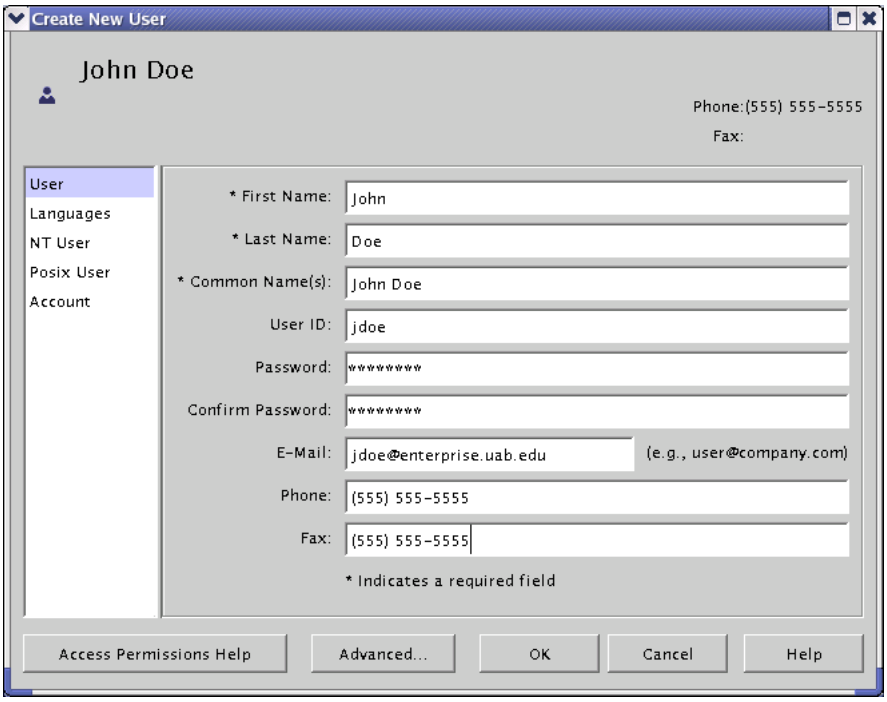
(22) Right click on the People Organizational Unit.



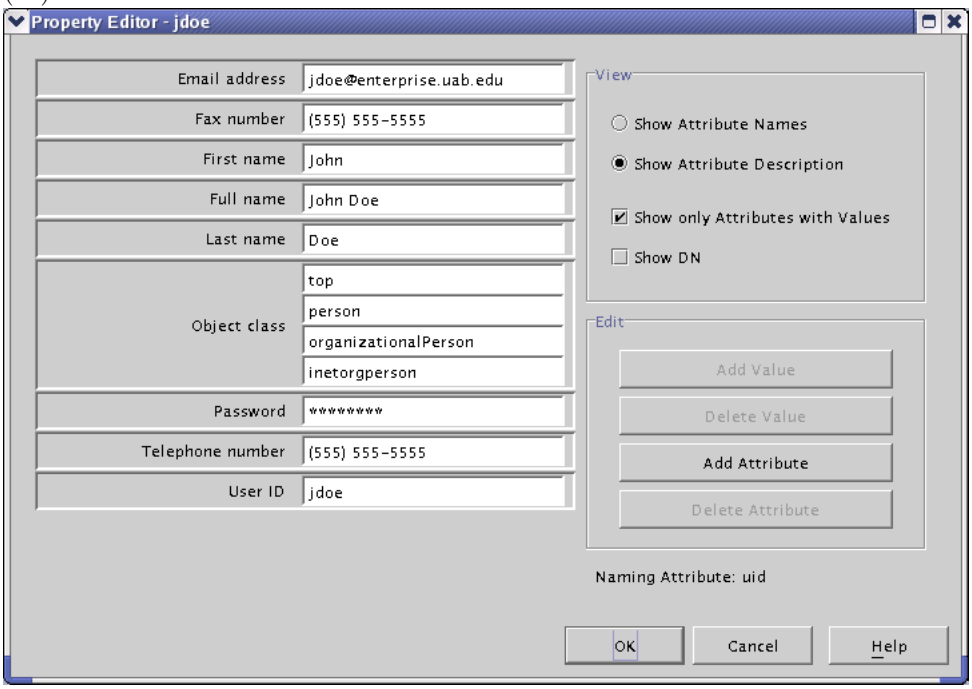
(23) Navigate accordingly to create a new user.



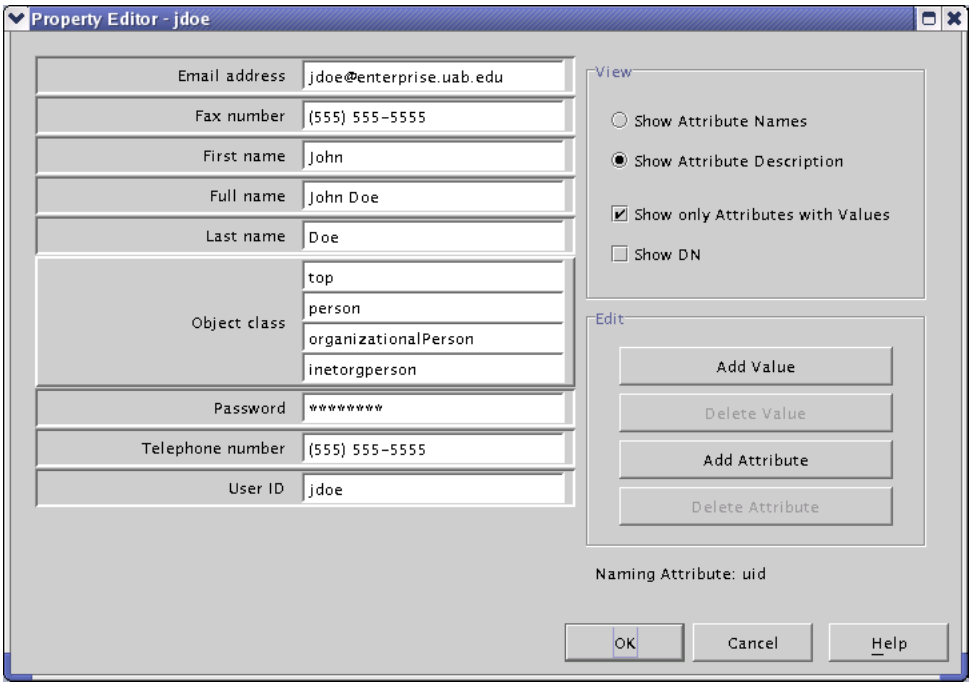
(24) Enter all necessary information into the form to create a new individual.



(25) Click on the Advanced... button.

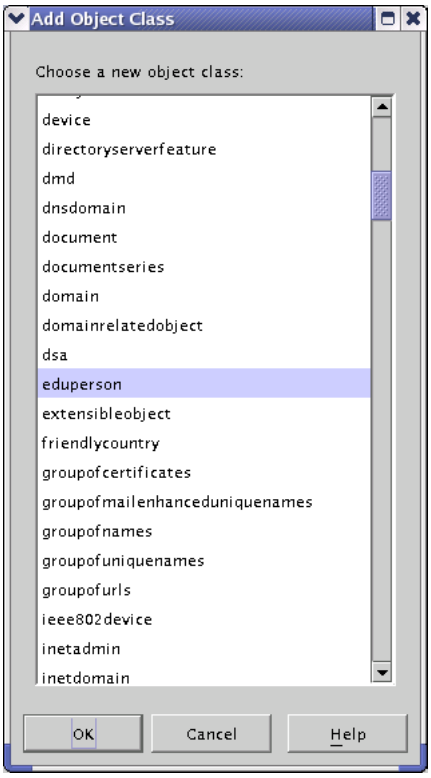


(26) Click on the Object Class attribute.



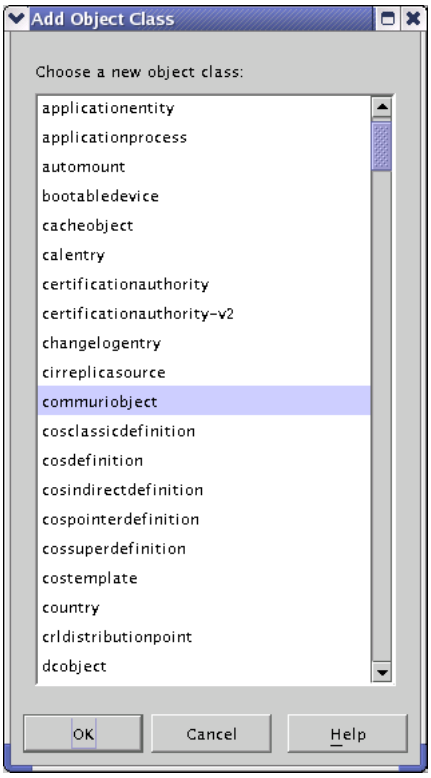
(27) Click on Add Value under the Edit heading.

(28) Add the EduPerson object class to the person by selecting the EduPerson object in the list and clicking on the Ok button.



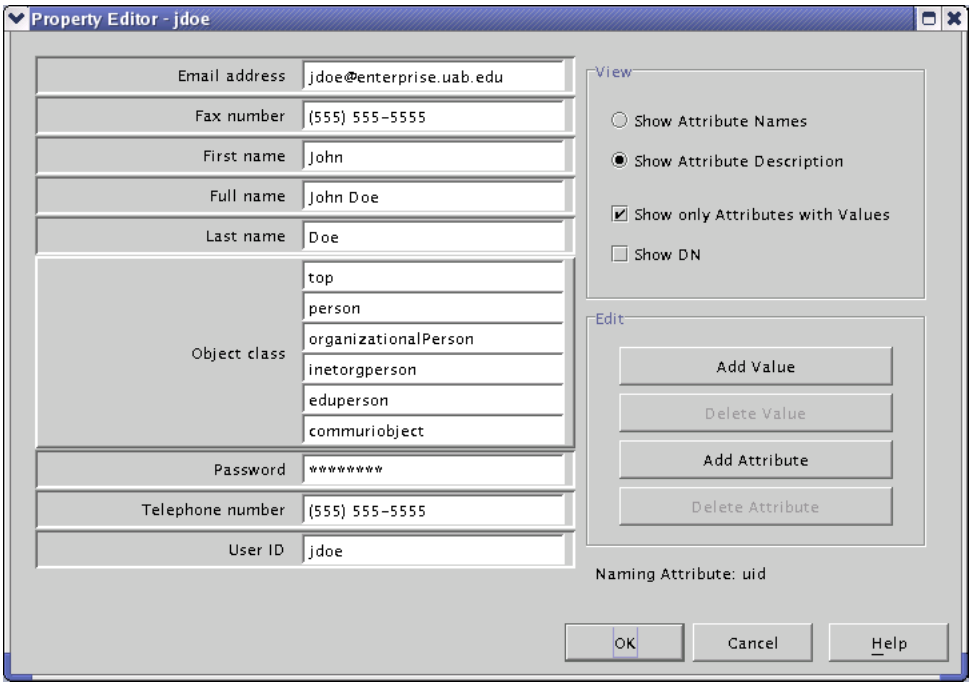
(29) Click on Add Value under the Edit heading again.

(30) Add the commURIObject object class to the person by selecting the commURIObject object in the list and clicking on the Ok button.

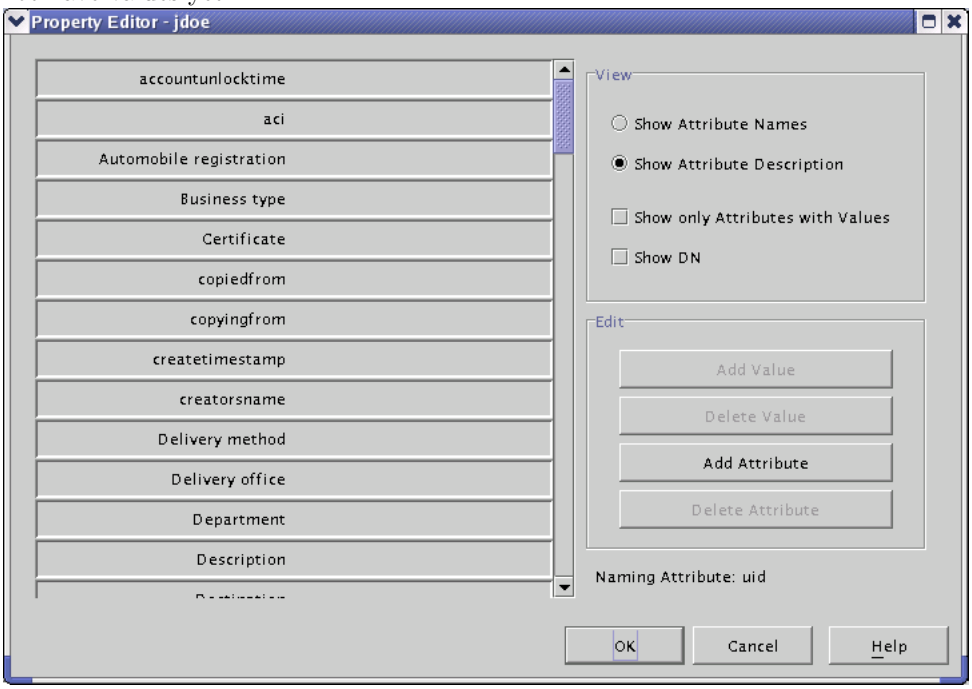


(31) If you need to add additional institutional specific objects to your newly created person, you can do so now.

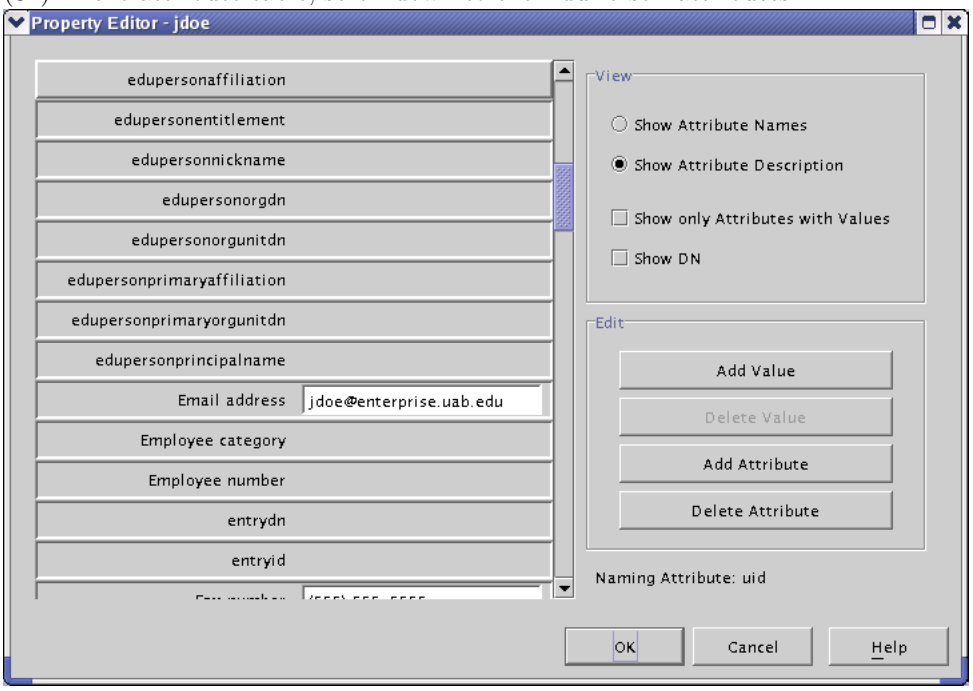
(32) You should now be able to see the object classes that you have added to the newly created person.



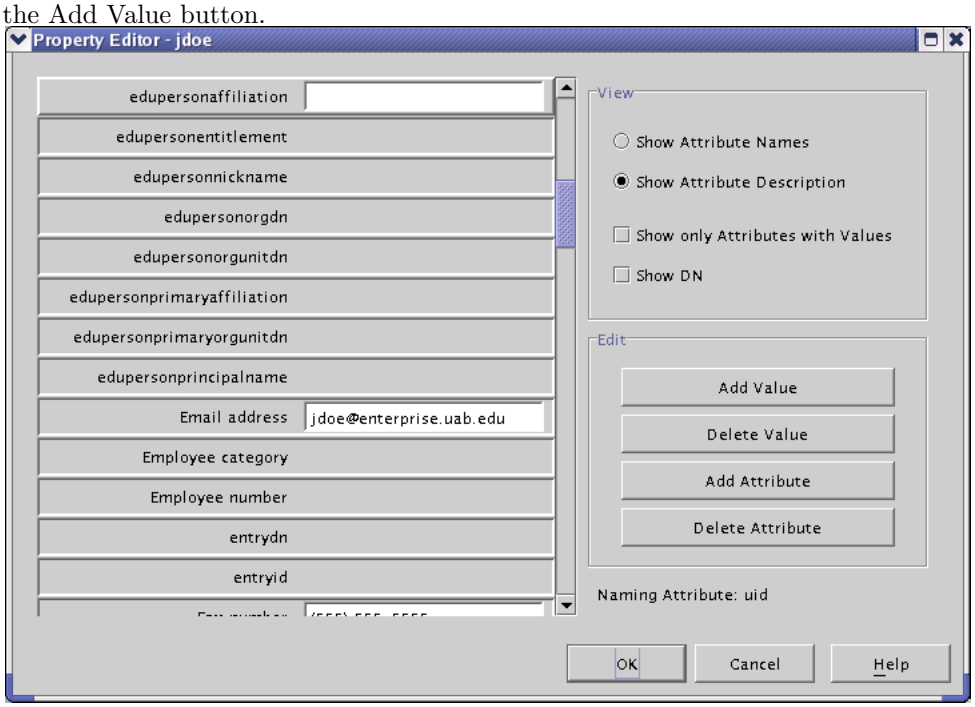
(33) Uncheck the Show only Attributes with Values checkbox in the user Property Editor window. This will allow you to view all of the attributes that do not have values yet.



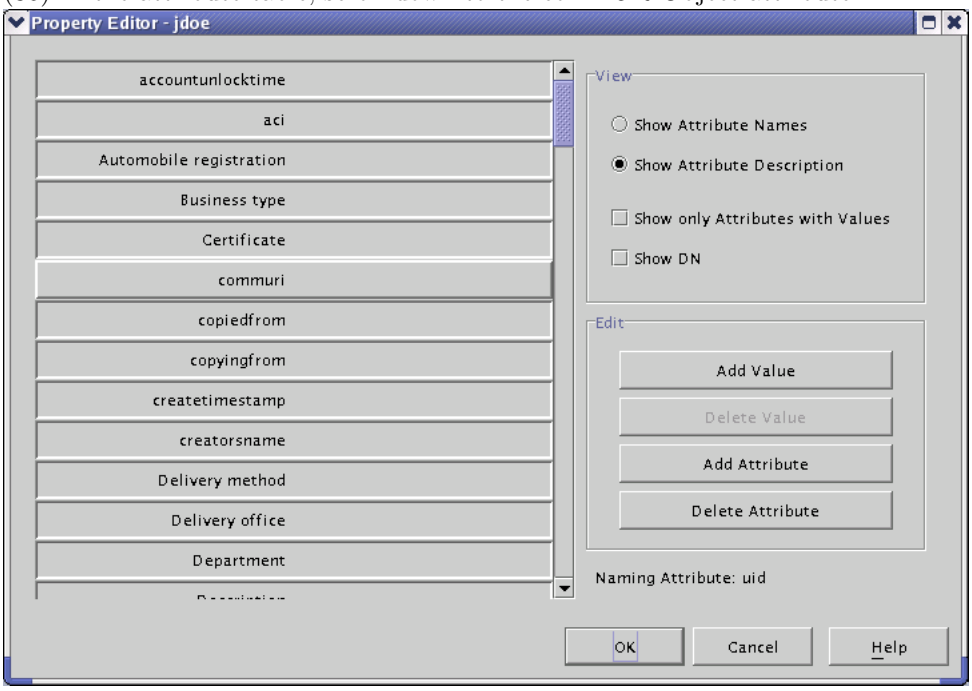
(34) In the attribute table, scroll down to the EduPerson attributes.



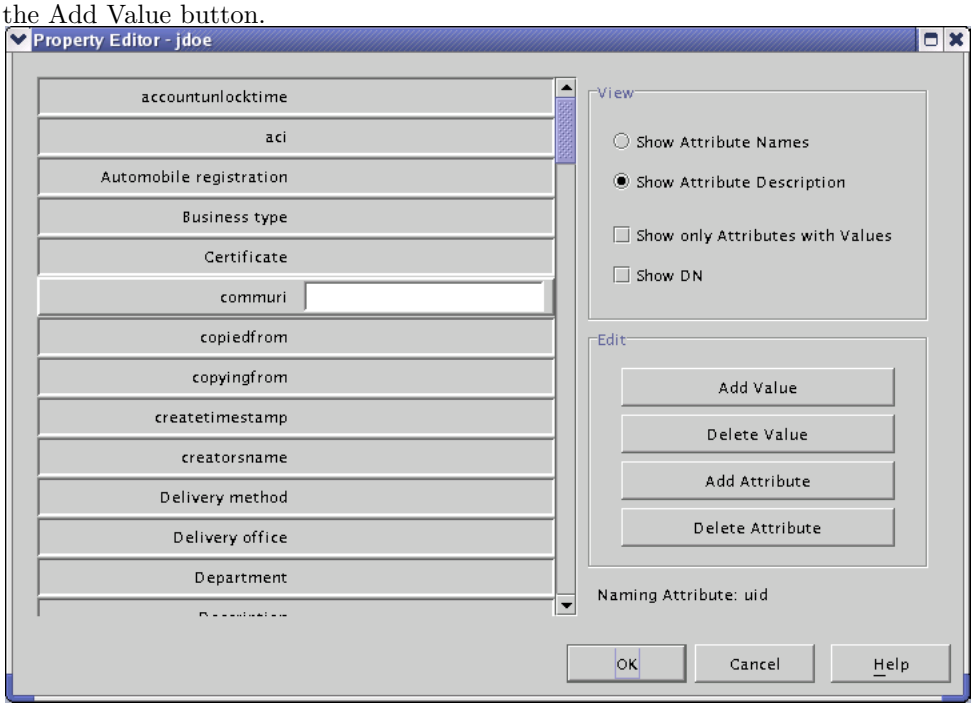
(35) Enter values for the attributes by clicking on the attribute and clicking on the Add Value button.



(36) In the attribute table, scroll down to the commURIObject attribute.



(37) Enter values for the attribute by clicking on the attribute and clicking on the Add Value button.



(38) Add values to any institution-specific attributes using the same method outlined in the previous steps.

(39) Click on the Ok button to close the Property Editor window.

(40) Click on the Ok button to close the Create New User window which will create the new user.

(41) Repeat this process to create additional users.

9.5 Populating an H.350 Directory with Sun ONE (iPlanet) Directory Server

9.5.1 Notes

None.

9.5.2 Requirements

- A running Sun ONE Directory Server on Solaris
- A running Sun ONE Administration Server on Solaris
- A version of the Sun ONE Administration Console on Solaris
- You should know the values for the following :
  - The Installation Directory for the Sun ONE Directory Server
  - The Directory Manager Distinguished Name
  - The Directory Manager Password
  - The Domain Name of the server
  - The port number on which the Directory Server is running
- You should be familiar with A Recipe for Configuring and Operating LDAP Directories.

9.5.3 Components

- Directory Server
  - This is the actual LDAP directory server.
- Administration Server
  - This is a common front-end to access the all iPlanet servers including the iPlanet Directory server.
- Administration Console
  - This is a common user interface which uses the administration server to communicate with the directory server.

9.5.4 Legend

Because of the differences in individual needs, certain parameters of the installed Sun ONE Directory Server may be quite different from the typical install. It is for this purpose that the following legend was created.

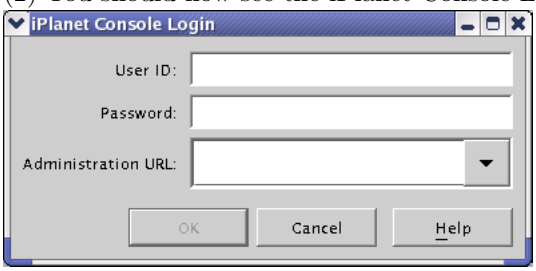
Parameter	Value
Sun ONE Directory Server Installation Directory	<ldap_dir>
Directory Manager Distinguished Name	<manager_dn>
Directory Manager Password	<manager_pw>
Fully Qualified Domain Name of the Server	<ldap_dns>
Sun ONE Directory Server Port Number	<ldap_port>

9.5.5 Instructions

(1) Run the following command from your Sun ONE Directory Server Installation Directory (<ldap_dir>).

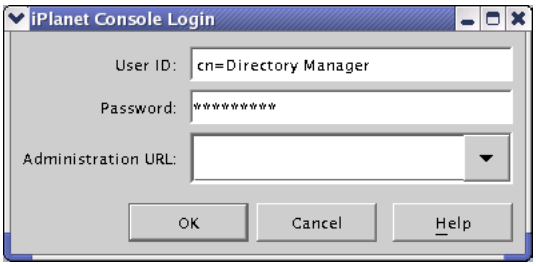
./startconsole

(2) You should now see the iPlanet Console Login window.

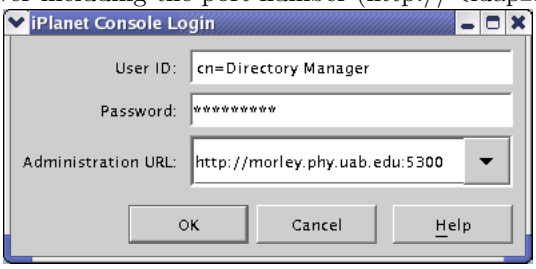


(3) In the User ID field, enter the Directory Manager Distinguished Name (<manager_dn>).

(4) In the Password field, enter the Directory Manager Password (<manager_pw>).

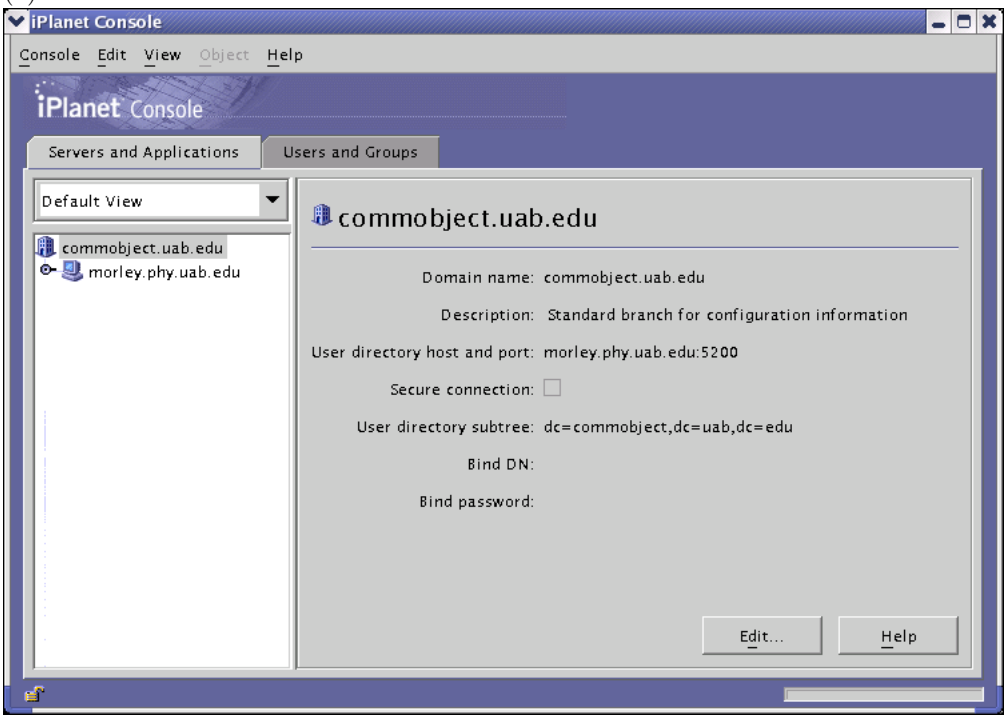


(5) In the Administration URL field, enter in the URL for the directory server including the port number (http://<ldap_dns>:<ldap_port>).



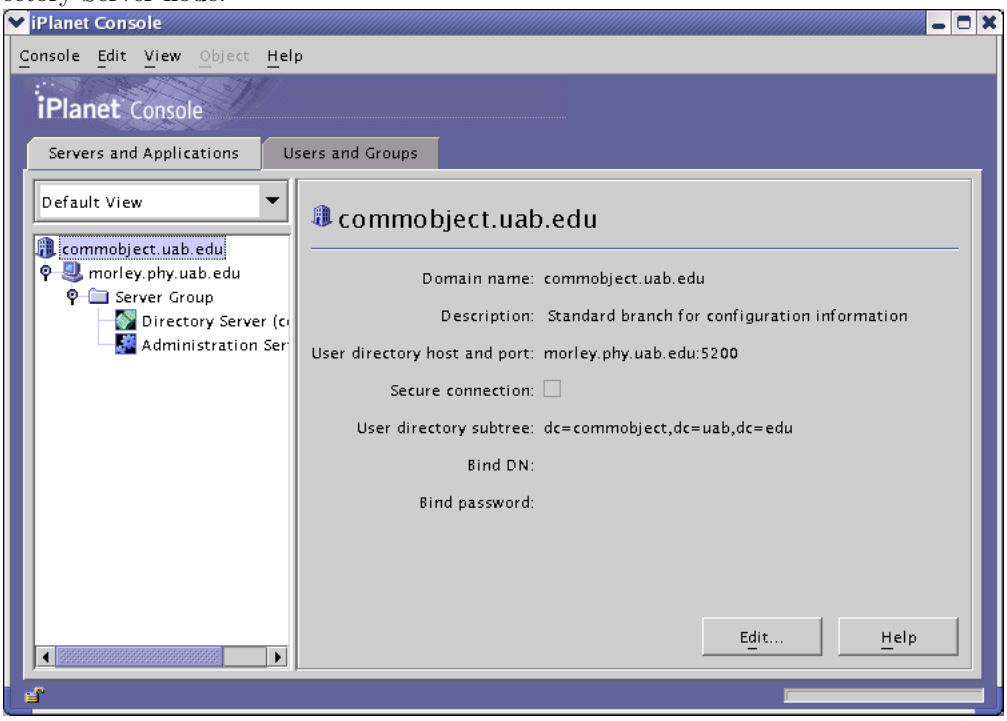
(6) Click OK.

(7) You should now see the iPlanet Console.

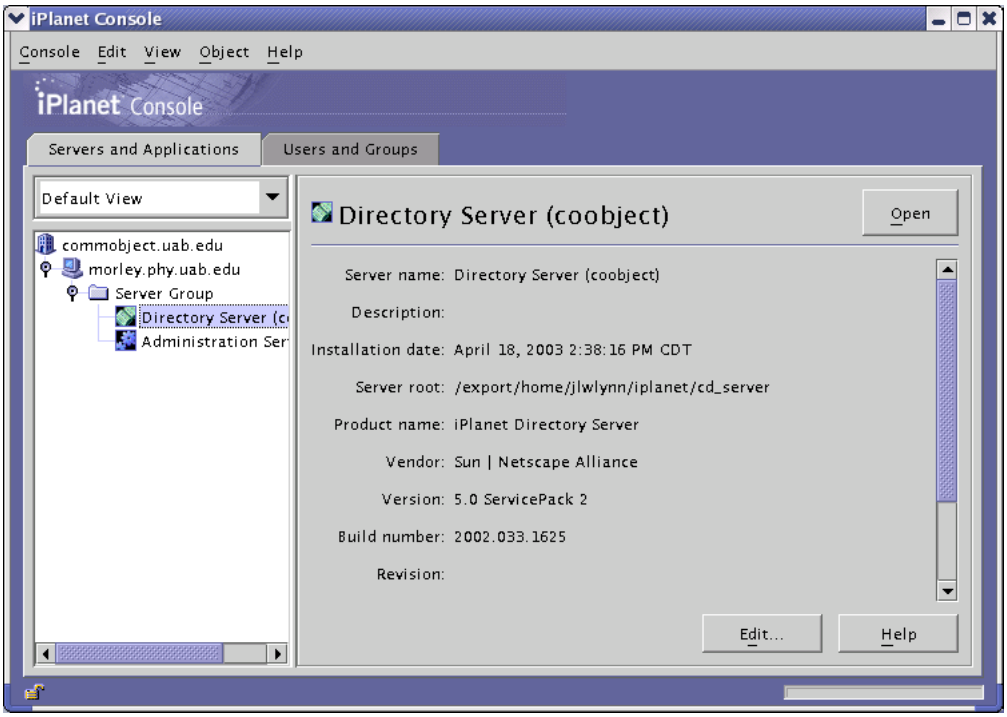


(8) On the left-hand side of the iPlanet Console Desktop, you should see a Java tree structure containing the name of your directory server and directly beneath it is the distinguished name of your server (<ldap_dns>).

(9) Click on the file handles in the tree to open them until you reach the Directory Server node.



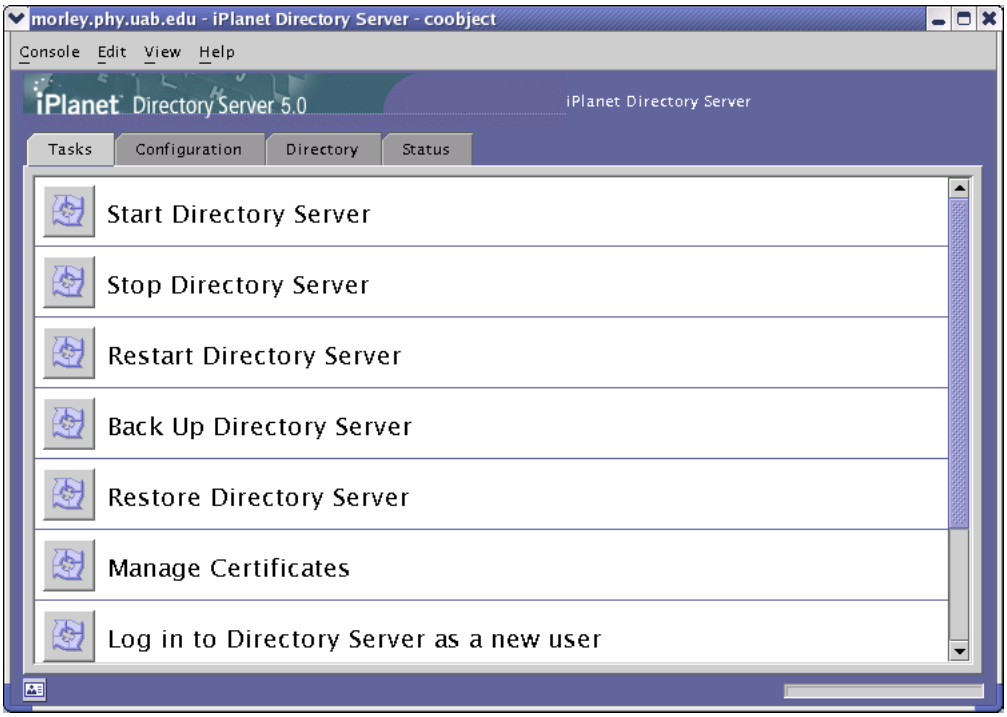
(10) Click on the Directory Server node.



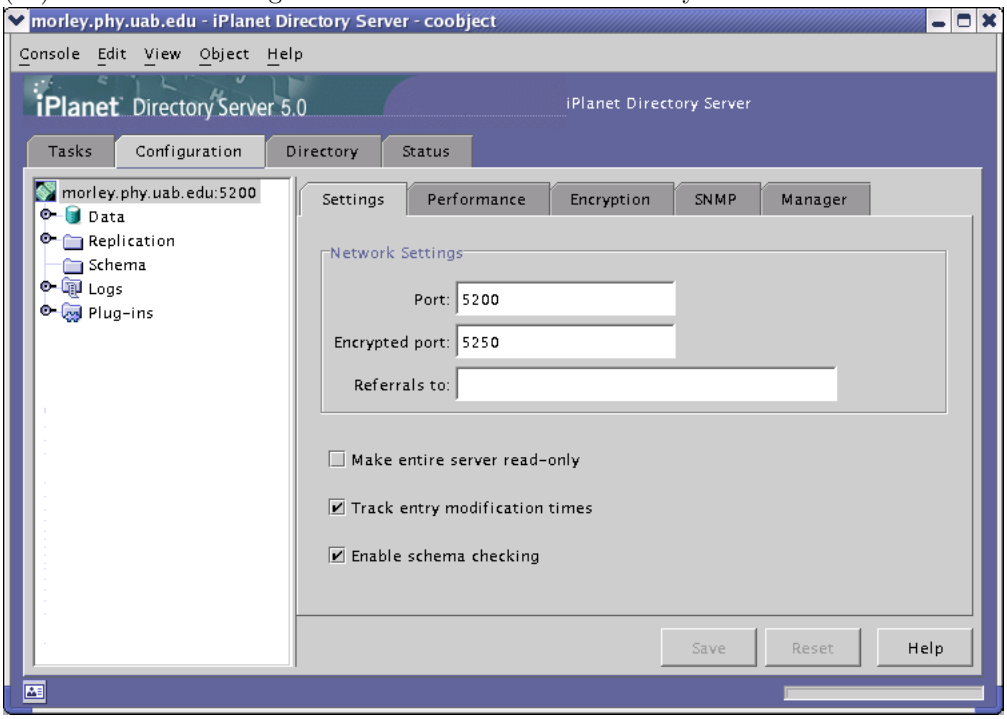
(11) You should now see some basic property information about your directory server on the right-hand side of the iPlanet Console Desktop.

(12) Click on the Open button contained in the property information..

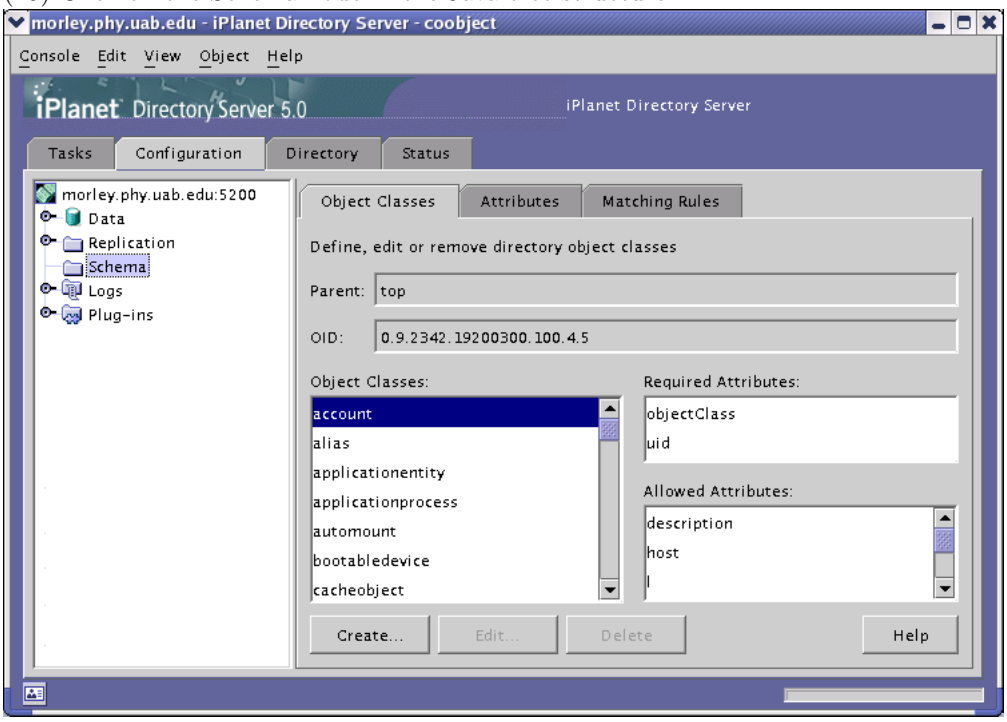
(13) You should now see your directory server open in a new iPlanet Directory Server window.



(14) Click on the Configuration tab in the iPlanet Directory Server window.

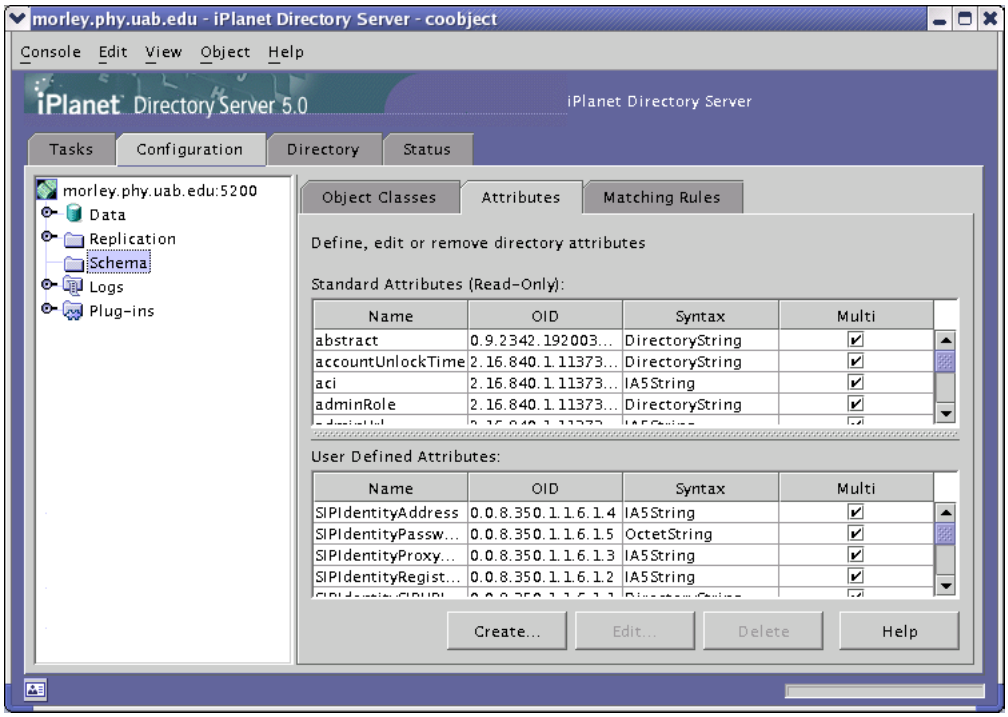


(15) Click on the Schema node in the Java tree structure.



(16) Click on the Attributes tab on the right-hand side of the iPlanet Directory Server Desktop.

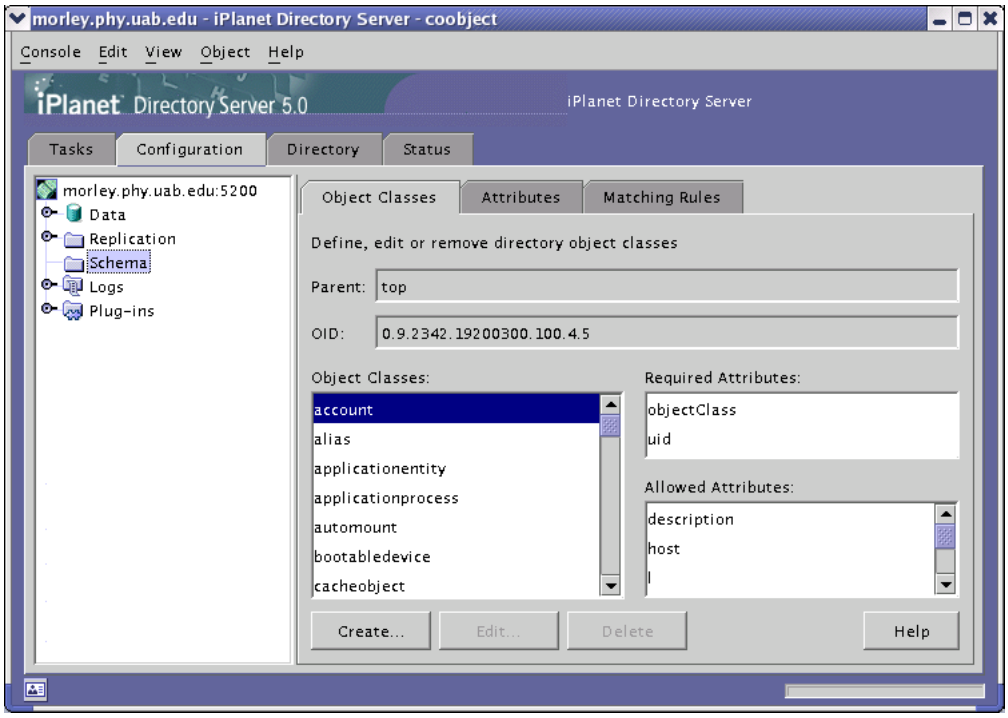
- Using the iPlanet Directory Server to create new objects, you must create the attributes, create the object, then add the attributes to the objects.



(17) Use the Create... button to add any additional attributes that your institution may need.

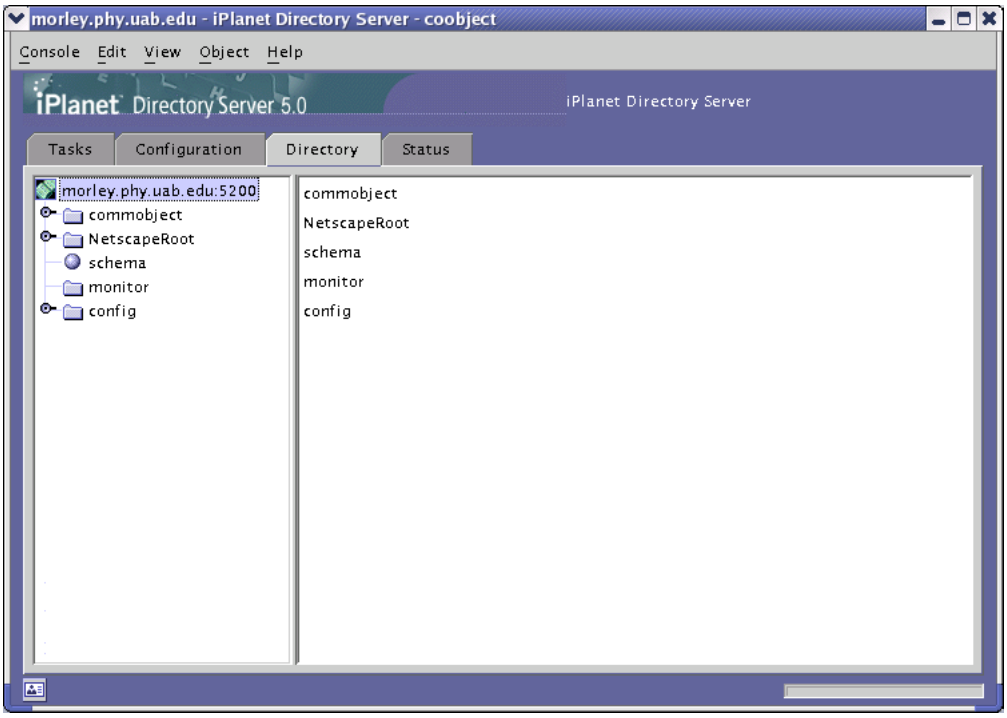
(18) Click on the Object Classes tab on the right-hand side of the iPlanet Directory Server Desktop.

- Using the iPlanet Directory Server to create new objects, you must create the attributes, create the object, then add the attributes to the objects.

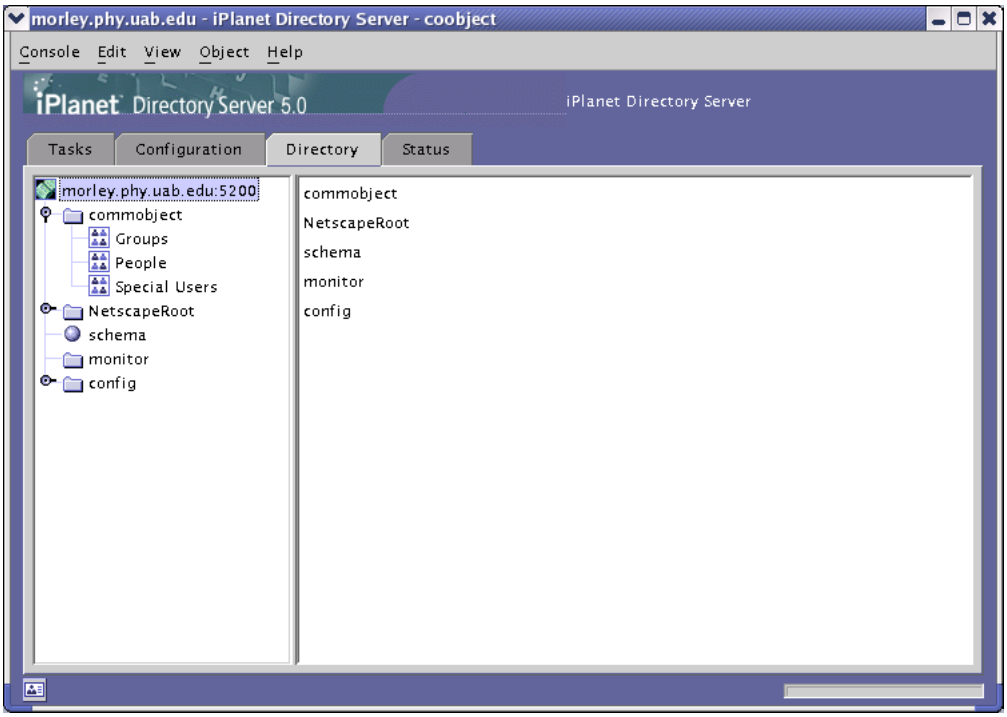


(19) Use the Create... button to add any additional object classes that your institution may need.

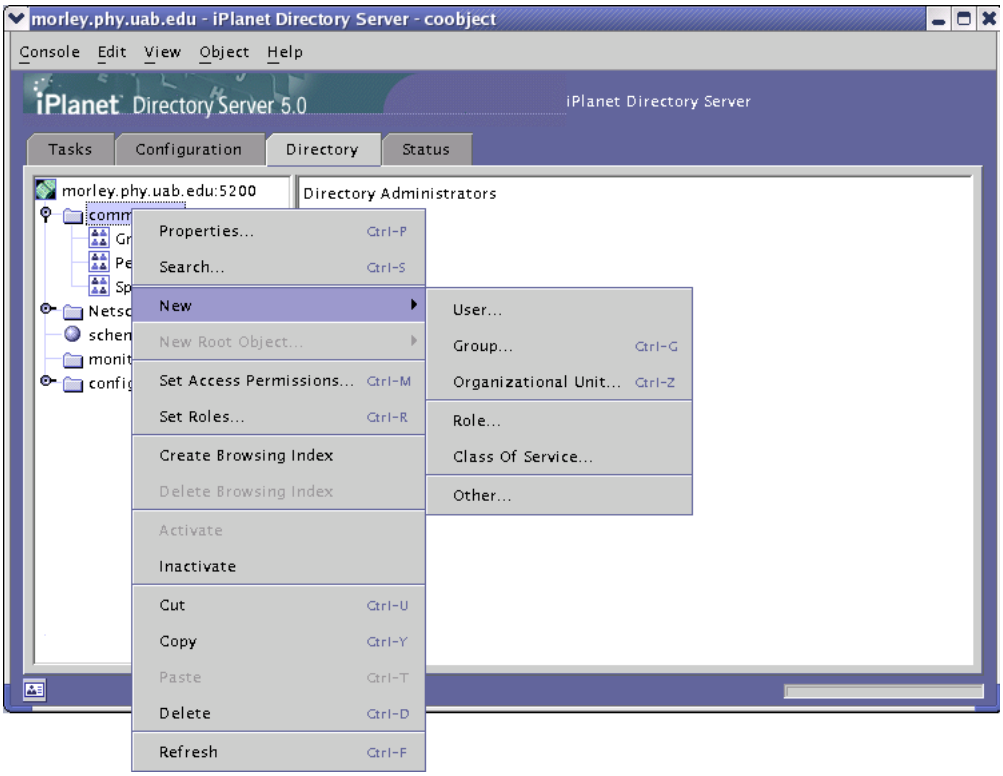
(20) Click on the Directory tab at the top of the iPlanet Directory Server Desktop.



(21) Open your directory data to reveal the organizational units already in place.

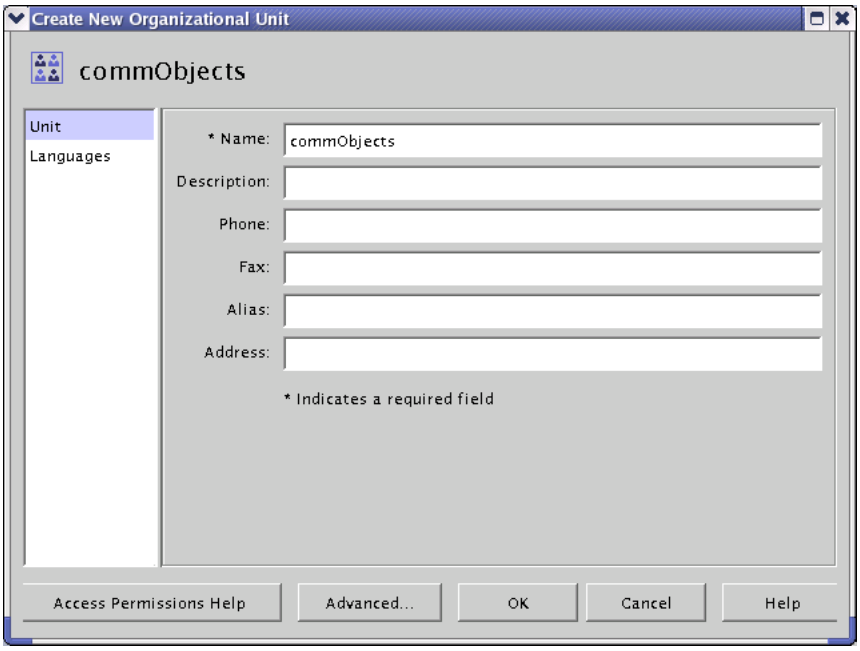


(22) Right-click on your directory root.



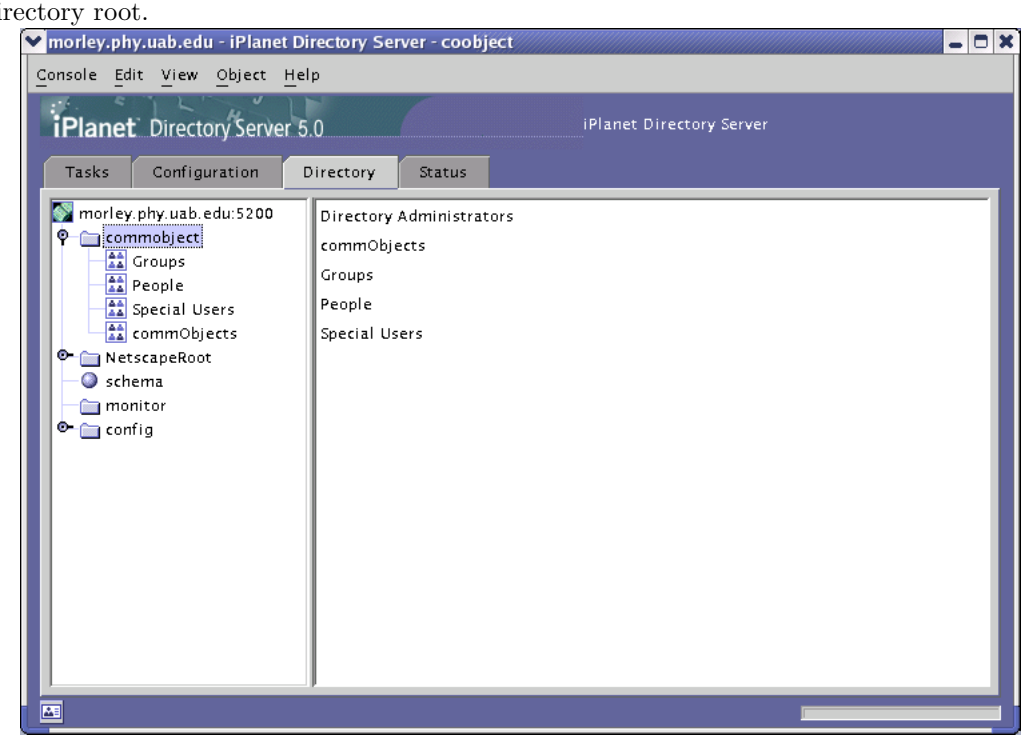
(23) Navigate accordingly to create a new Organizational Unit.

(24) Enter all necessary information into the form to create a new Organizational Unit for your *commObjects*.

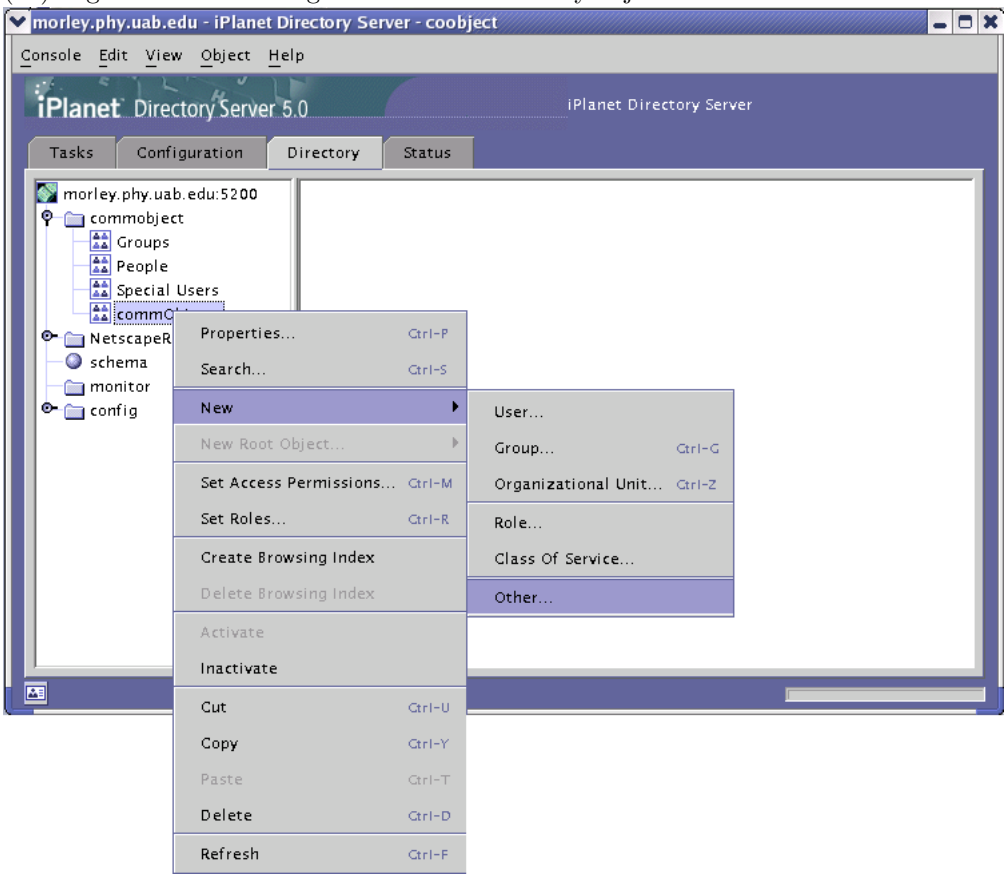


(25) Click the OK button.

(26) You should now be able to see the new Organizational Unit under your directory root.

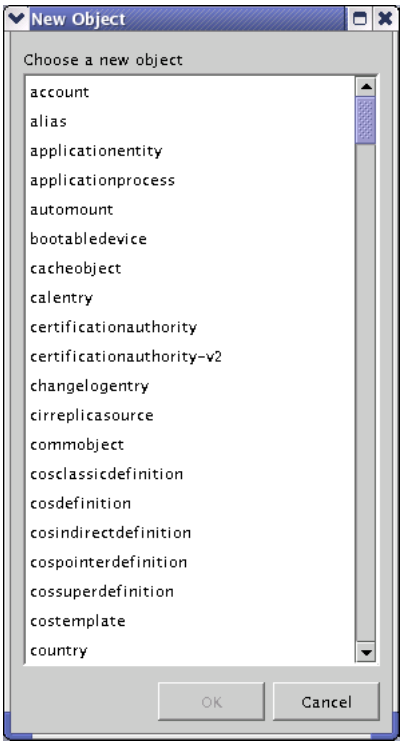


(27) Right-click on the Organizational Unit that you just created.

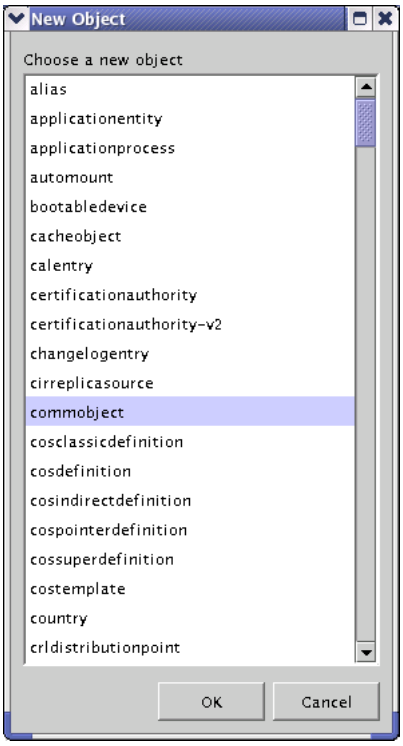


(28) Navigate accordingly to create a new object by clicking on the Other menu item. This will allow you to create a new *commObject* object.

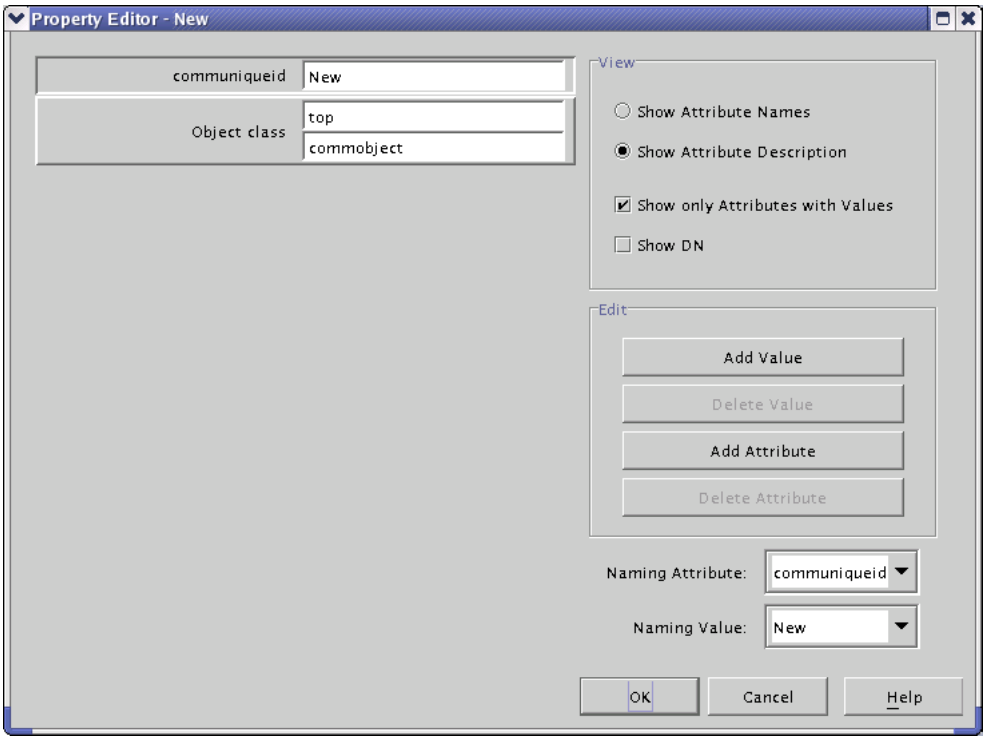
(29) You should now see the New Object window.



(30) Scroll down in the list to choose *'commObject'*.

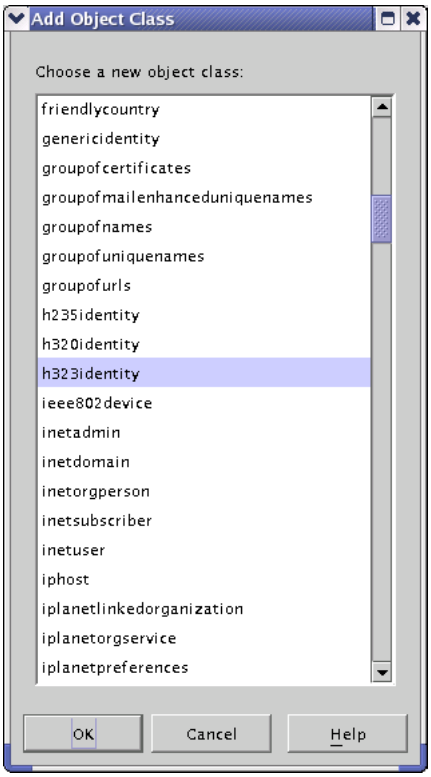


(31) Click the OK button.
(32) Click on the *'Object class'* attribute.

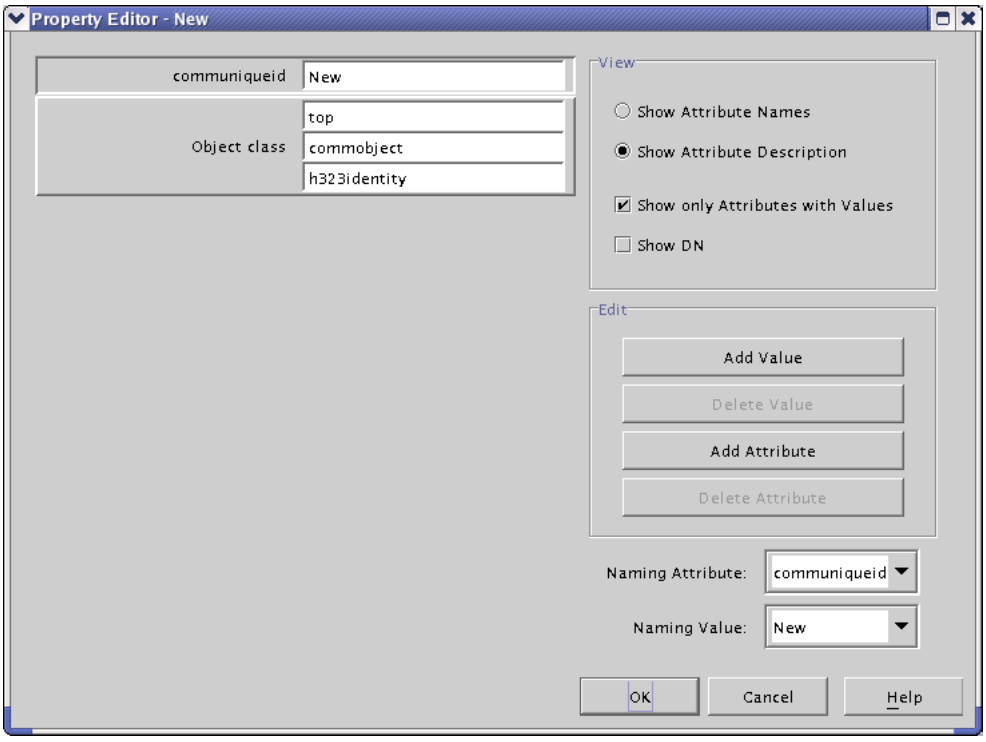


(33) Click on the *'Add Value'* button. You should now see the *'Add Object Class'* window.

(34) Add Auxiliary classes (in this case, *h323Identity*) to match your endpoint by choosing them in the list and clicking on the *'OK'* button.

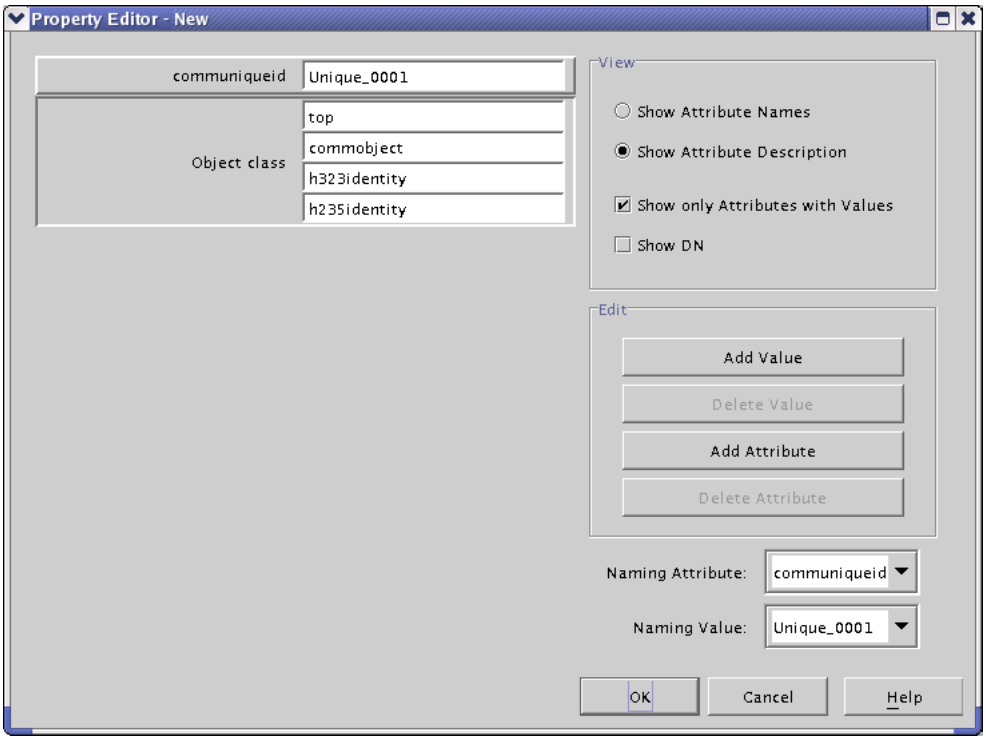


(35) You should now see the updated *'Object class'* attribute in the Property Editor screen.

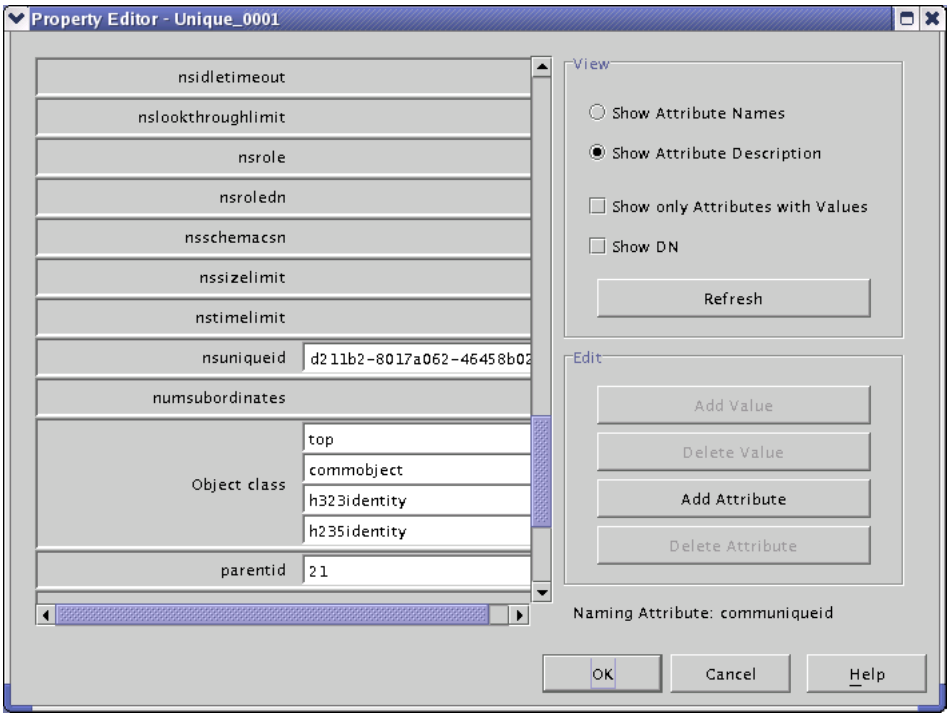


(36) Continue this process until you have added all of the auxiliary classes that you need for this endpoint.

(37) Click on the *'communiqueid'* attribute and alter the current value to provide a unique identification for this endpoint.

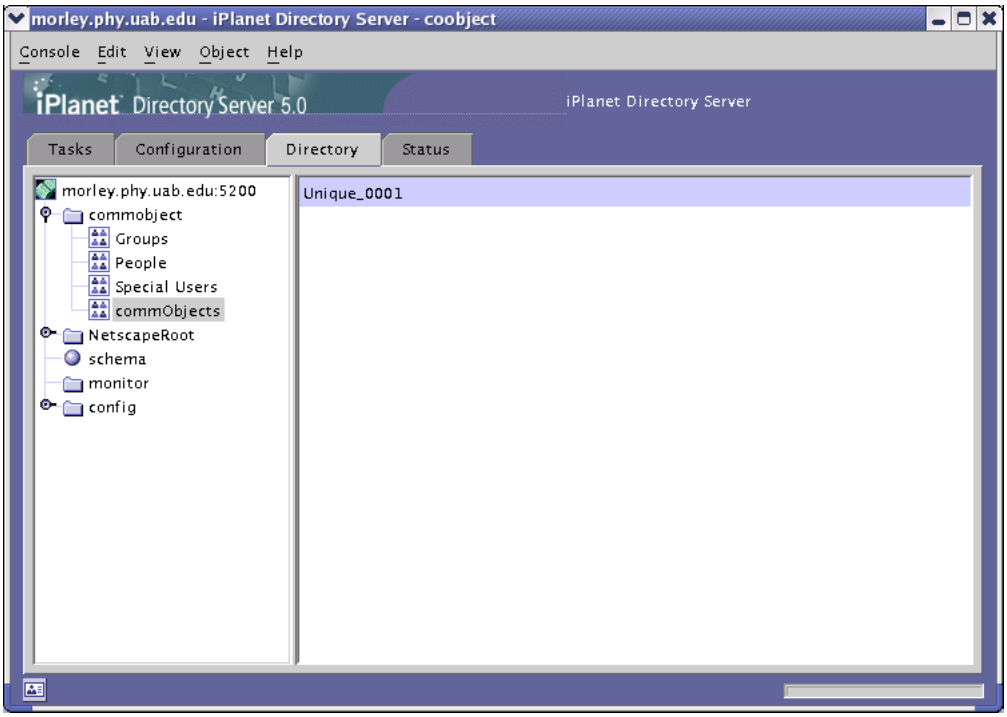


(38) Uncheck *'Show only Attributes with Values'* checkbox to add values to other attributes.



(39) Click the OK button to create the *commObject*.

(39) You should now be able to see your newly created *commObject* under the Directory tab.



10 Installing and Configuring your Directory Server: OpenLDAP Directory Server

10.1 Installing the OpenLDAP Directory Server

10.1.1 Notes

Detailed documents can always be found at the OpenLDAP website. These instructions work successfully on Redhat 8.0; your platform may differ.

10.1.2 Requirements

- UNIX (or UNIX-like system)
- A C Development Environment
- POSIX REGEX routines
- Berkeley Networking (socket/select) routines
- Berkeley Database 4.1 from Sleepycat Software
- You should be familiar with A Recipe for Configuring and Operating LDAP Directories.

10.1.3 Components

- Directory Server
  - This is the actual LDAP directory server.

10.1.4 Installation Decisions

There are numerous decisions one can make prior to installing the OpenLDAP server. Each installation decision is carefully defined on the OpenLDAP documentation webpage as well as in the configure script.

10.1.5 Installation

(1) Download the latest Berkeley Database 4.1 from Sleepycat Software in tar and gziped format.

(2) Expand the installation file.

Example: tar xvfz db-4.1.25.tar.gz

(3) If you are not already the root user, please switch to the root user.

(4) Change your current working directory to be the newly created directory.

Example: cd db-4.1.25

(5) Change your current working directory to the UNIX build directory.

cd build_unix

(6) View and choose the configuration options for the Berkeley Database application.

../dist/configure –help

(7) Run the configuration utility for the Berkeley Database application specifying the configuration options you wish to include. It is preferable to at least include the '–prefix=/usr/local' option as OpenLDAP will find the library files by default without any specification.

../dist/configure –prefix=/usr/local

(8) Build the Berkeley Database application.

make

(9) Install the Berkeley Database application.

make install

(10) Download the latest OpenLDAP Directory Server file for Linux.

(11) Expand the installation file.

Example: tar xvfz openldap-2.1.21

(12) If you are not already the root user, please switch to the root user.
(13) Change your current working directory to be the newly created directory.

Example: cd openldap-2.1.21

(14) View and choose the configuration options for the OpenLDAP application.

./configure –help

(15) Run the configuration utility for the OpenLDAP application specifying the configuration options you wish to include.

Example: env CC=gcc ./configure –enable-ldbm \ –with-cyrus-sasl –with-tls

(16) Build the dependencies for the OpenLDAP application.

make depend

(17) Build the OpenLDAP application.

make

(18) If the system is not built, review and alter your configuration settings and rebuild.
(19) Test the OpenLDAP application.

make test

(20) If the tests fail, review and alter your configuration settings and rebuild.
(21) Install the OpenLDAP application.

make install

(22) OpenLDAP should now be installed.

10.2 Configuring an Enterprise Directory with OpenLDAP Directory Server

10.2.1 Notes

None.

10.2.2 Requirements

- A running OpenLDAP Directory Server on Linux
- Knowledge of the following values :
  - The Configuration Directory for the OpenLDAP Directory Server
  - The Schema Directory for the OpenLDAP Directory Server
  - The Executable Directory for the OpenLDAP Directory Server
  - The Domain Name of the server
- Familiarity with A Recipe for Configuring and Operating LDAP Directories

10.2.3 Components

- Directory Server
  - This is the actual LDAP directory server.
- EduPerson Schema File
  - This is the file that contains the schema changes necessary to support the EduPerson object.
- commURI Schema File
  - This is the file that contains the schema changes necessary to support the commURI object.

10.2.4 Legend

Because of the differences in individual needs, certain parameters of the installed Sun ONE Directory Server may be quite different from the typical install. It is for this purpose that the following legend was created.

Parameter	Value	Example
OpenLDAP Configuration Directory	<config_dir>	/usr/local/etc/openldap/
OpenLDAP Schema Directory	<schema_dir>	/usr/local/etc/openldap/schema/
OpenLDAP Executable Directory	<exe_dir>	/usr/local/libexc/
Enterprise slapd.conf File	<slapd_file>	/usr/local/etc/openldap/slapd.conf.enterprise
Fully Qualified Domain Name of the OpenLDAP Server	<ldap_dns>	enterprise.uab.edu
OpenLDAP Directory Server Port Number	<ldap_port>	389

10.2.5 Instructions

(1) Download the latest EduPerson Schema file from here and save it in the schema directory, <schema_dir>.

(2) Download the latest commURI Schema file from here and save it in the schema directory, <schema_dir>.

(3) Create a copy (<slapd_file>) of the <config_dir>/slapd.conf file.

cp <config_dir>/slapd.conf <config_dir>/slapd.conf.enterprise

(4) Open <slapd_file> with a text editor.

(5) Add an entry for the inetOrgPerson Schema file as an include directive in the global section of the file.

include /usr/local/etc/openldap/schema/inetorgperson.schema

(6) Add an entry for the EduPerson Schema file as an include directive in the global section of the file.

include /usr/local/etc/openldap/schema/eduperson.schema

(7) Add an entry for the commURI Schema file as an include directive in the global section of the file.

include /usr/local/etc/openldap/schema/commURI.schema

(8) Change the pidfile directive to a unique file among other slapd instances.

pidfile /usr/local/var/slapd.pid.enterprise

(9) Change the argsfile directive to a unique file among other slapd instances.

argsfile /usr/local/var/slapd.args.enterprise

(10) Change the suffix directive to reflect the base DN of your directory.

suffix "dc=enterprise,dc=uab,dc=edu"

(11) Change the rootdn of the directory to reflect the DN of the directory administrator.

rootdn "cn=admin,dc=enterprise,dc=uab,dc=edu"

(12) Use the slappasswd command line application to create a password for the rootpw directive in the file. You should generally not use cleartext passwords.

rootpw {SSHA}kDKWEwSNLVvckbOEH+TBRvEqOQtkUMJy

(13) Change the directory directive to a unique directory among other slapd instances. This directory must exist prior to running slapd.

directory /usr/local/var/openldap-data-enterprise

(14) Set indexing attributes.

index default pres,eq index uid index objectClass eq

(15) Make other optional changes. * Add support for SASL * Add support for TLS/SSL * Add logging support
(16) Start slapd with the following arguments.

<exe_dir>/slapd -f <slapd_file> -h "ldap://<ldap_dns>:<ldap_port>/"

10.3 Configuring a *commObject* Directory with OpenLDAP Directory Server

10.3.1 Notes

None.

10.3.2 Requirements

- A running OpenLDAP Directory Server on Linux
- Knowledge of the following values :
  - The Configuration Directory for the OpenLDAP Directory Server
  - The Schema Directory for the OpenLDAP Directory Server
  - The Executable Directory for the OpenLDAP Directory Server
  - The Domain Name of the server
- Familiarity with A Recipe for Configuring and Operating LDAP Directories

10.3.3 Components

- Directory Server
  - This is the actual LDAP directory server.
- commObject Schema File
  - This is the file that contains the schema changes necessary to support the *commObject* object.
- genericIdentity Schema File
  - This is the file that contains the schema changes necessary to support the *genericIdentity* object.

- h235Identity Schema File
  - This is the file that contains the schema changes necessary to support the *h235Identity* object.
- h320Identity Schema File
  - This is the file that contains the schema changes necessary to support the *h320Identity* object.
- h323Identity Schema File
  - This is the file that contains the schema changes necessary to support the *h323Identity* object.
- sipIdentity Schema File
  - This is the file that contains the schema changes necessary to support the *sipIdentity* object.

10.3.4 Legend

Because of the differences in individual needs, certain parameters of the installed Sun ONE Directory Server may be quite different from the typical install. It is for this purpose that the following legend was created.

Parameter	Value	Example
OpenLDAP Configuration Directory	<config_dir>	/usr/local/etc/openldap/
OpenLDAP Schema Directory	<schema_dir>	/usr/local/etc/openldap/schema/
OpenLDAP Executable Directory	<exe_dir>	/usr/local/libexc/
commObject slapd.conf File	<slapd_file>	/usr/local/etc/openldap/slapd.conf.commobject
Fully Qualified Domain Name of the OpenLDAP Server	<ldap_dns>	commobject.uab.edu
OpenLDAP Directory Server Port Number	<ldap_port>	389

10.3.5 Instructions

(1) Download the latest *commObject* Schema file from here and save it in the schema directory, <schema_dir>.

(2) Download the latest *genericIdentity* Schema file from here and save it in the schema directory, <schema_dir>.

(3) Download the latest *h235Identity* Schema file from here and save it in the schema directory, <schema_dir>.

(4) Download the latest *h320Identity* Schema file from here and save it in the schema directory, <schema_dir>.

(5) Download the latest *h323Identity* Schema file from here and save it in the schema directory, <schema_dir>.

(6) Download the latest *sipIdentity* Schema file from here and save it in the schema directory, <schema_dir>.

(7) Create a copy (<slapd_file>) of the <config_dir>/slapd.conf file.

cp <config_dir>/slapd.conf <config_dir>/slapd.conf.commobject

(8) Open <slapd_file> with a text editor.

(9) Add an entry for the *commObject* Schema file as an include directive in the global section of the file.

include /usr/local/etc/openldap/schema/commObject.schema

(10) Add an entry for the *genericIdentity* Schema file as an include directive in the global section of the file.

include /usr/local/etc/openldap/schema/genericIdentity.schema

(11) Add an entry for the *h235Identity* Schema file as an include directive in the global section of the file.

include /usr/local/etc/openldap/schema/h235Identity.schema

(12) Add an entry for the *h320Identity* Schema file as an include directive in the global section of the file.

include /usr/local/etc/openldap/schema/h320Identity.schema

(13) Add an entry for the *h323Identity* Schema file as an include directive in the global section of the file.

include /usr/local/etc/openldap/schema/h323Identity.schema

(14) Add an entry for the *sipIdentity* Schema file as an include directive in the global section of the file.

include /usr/local/etc/openldap/schema/sipIdentity.schema

(15) Change the pidfile directive to a unique file among other slapd instances.

pidfile /usr/local/var/slapd.pid.commobject

(16) Change the argsfile directive to a unique file among other slapd instances.

argsfile /usr/local/var/slapd.args.commobject

(17) Change the suffix directive to reflect the base DN of your directory.

suffix ”dc=commobject,dc=uab,dc=edu”

(18) Change the rootdn of the directory to reflect the DN of the directory administrator.

rootdn ”cn=admin,dc=commobject,dc=uab,dc=edu”

(19) Use the slappasswd command line application to create a password for the rootpw directive in the file. You should generally not use cleartext passwords.

rootpw {SSHA}kDKWEwSNLVvckbOEH+TBRvEqOQtkUMJy

(20) Change the directory directive to a unique directory among other slapd instances. This directory must exist prior to running slapd.

directory /usr/local/var/openldap-data-commobject

(21) Set indexing attributes.

index default pres,eq index commUniqueId index objectClass eq

(22) Make other optional changes.

- Add support for SASL
- Add support for TLS/SSL
- Add logging support

(23) Start slapd with the following arguments.

<exe_dir>/slapd -f <slapd_file> -h ”ldap://<ldap_dns>:<ldap_port>/”

10.4 Populating an Enterprise Directory with OpenLDAP Directory Server

10.4.1 Notes

OpenLDAP uses LDIF files to add entries to the Directory Server. Therefore, to populate the directory, you must first create an LDIF file with entries to add to the server.

10.4.2 Requirements

- A running OpenLDAP Directory Server on Linux
- You should be familiar with A Recipe for Configuring and Operating LDAP Directories.

10.4.3 Components

- Directory Server
  - This is the actual LDAP directory server.

10.4.4 Legend

Because of the differences in individual needs, certain parameters of the installed Sun ONE Directory Server may be quite different from the typical install. It is for this purpose that the following legend was created.

Parameter	Value	Example
The fully qualified domain name of the server on which OpenLDAP is running	<ldap_host>	enterprise.uab.edu
The port on which the OpenLDAP Directory Server is running	<ldap_port>	389
The Distinguished Name for the root user as defined in <slapd_file>	<root_dn>	cn=admin,dc=enterprise,dc=uab,dc=edu
The password for the root user as defined in <slapd_file>	<root_pw>	secret
The fully qualified path to the LDIF file that you are creating	<ldif_file>	/tmp/enterprise.ldif
The slapd.conf file which defines your Enterprise directory	<slapd_file>	/usr/local/etc/openldap/slapd.conf.enterprise

10.4.5 Instructions

(1) Create and open a file (<ldif_file>) with a text editor.
(2) Create your base DN in the text file.

```
# Base DN
#
dn: dc=enterprise,dc=uab,dc=edu
objectClass: top
objectClass: dcObject
objectClass: domain
dc: enterprise
```

(3) Create an organizational unit to contain users by appending to the file.

```
# People OU
#
dn: ou=people,dc=enterprise,dc=uab,dc=edu
ou: people
objectClass: top
objectClass: organizationalUnit
```

(4) Create the user entries by appending to the file.

```
# John Doe's Entry
#
dn: uid=jdoe,ou=people,dc=enterprise,dc=uab,dc=edu
cn: jdoe
cn: John Doe
givenname: John
sn: Doe
mail: jdoe@nowhere.com
userPassword: secret
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
objectClass: eduperson
telephonenumber: (555) 555-5555
facsimiletelephonenumber: (555) 555-5555
uid: jdoe
```

(5) Add the entries to the LDAP server with any of the following two methods.

With slapd running:

```
ldapadd -x -h <ldap_host> -p <ldap_port> -D <root_dn>
\ -w <root_pw> -f <ldif_file>
```

With slapd not running:

```
slapadd -f <slapd_file> -l <ldif_file>
```

10.5 Populating an H.350 Directory with OpenLDAP Directory Server

10.5.1 Notes

OpenLDAP uses LDIF files to add entries to the Directory Server. Therefore, to populate the directory, you must first create an LDIF file with entries to add to the server.

10.5.2 Requirements

- A running OpenLDAP Directory Server on Linux
- You should be familiar with A Recipe for Configuring and Operating LDAP Directories

10.5.3 Components

- Directory Server
  - This is the actual LDAP directory server.

10.5.4 Legend

Because of the differences in individual needs, certain parameters of the installed Sun ONE Directory Server may be quite different from the typical install. It is for this purpose that the following legend was created.

Parameter	Value	Example
The fully qualified domain name of the server on which OpenLDAP is running	<ldap_host>	commobject.uab.edu
The port on which the OpenLDAP Directory Server is running	<ldap_port>	389
The Distinguished Name for the root user as defined in <slapd_file>	<root_dn>	cn=admin,dc=commobject,dc=uab,dc=edu
The password for the root user as defined in <slapd_file>	<root_pw>	secret
The fully qualified path to the LDIF file that you are creating	<ldif_file>	/tmp/commobject.ldif
The slapd.conf file which defines your *commObject* directory	<slapd_file>	/usr/local/etc/openldap/slapd.conf.commobject

10.5.5 Instructions

(1) Create and open a file with a text editor.
(2) Create your base DN in the text file.

```
# Base DN
#
```

```
dn: dc=commobject,dc=uab,dc=edu
objectClass: top
objectClass: dcObject
objectClass: domain
dc: commobject
```

(3) Create an organizational unit to contain *commObjects* by appending to the file.

```
# commObject OU
#
dn: ou=commObjects,dc=commobject,dc=uab,dc=edu
ou: commObjects
objectClass: top
objectClass: organizationalUnit
```

(4) Create the *commObject* entries by appending to the file.

```
# John's commObject Entry
#
dn: commUniqueId=uniqueid_0001,ou=commObjects,dc=commobject,dc=uab,dc=edu
commUniqueId: uniqueid_0001
objectClass: top
objectClass: commObject
objectClass: h323Identity
objectClass: h235Identity
```

(5) Add the entries to the LDAP server with any of the following two methods.

With slapd running:

```
ldapadd -x -h <ldap_host> -p <ldap_port> -D <root_dn>
\ -w <root_pw> -f <ldif_file>
```

With slapd not running:

```
slapadd -f <slapd_file> -l <ldif_file>
```

11 Installing and Configuring your Directory Server: Microsoft Active Directory

11.1 Windows 2000 Server

11.1.1 Installing the Active Directory Server

11.1.2 Configuring an Enterprise Directory with Active Directory Server

11.1.3 Configuring a *commObject* Directory with Active Directory Server

11.1.4 Populating an Enterprise Directory with Active Directory Server

11.1.5 Populating an H.350 Directory with Active Directory Server

11.2 Windows 2003 Server

11.2.1 Installing the Active Directory Server

11.2.2 Configuring an Enterprise Directory with Active Directory Server

11.2.3 Configuring a *commObject* Directory with Active Directory Server

11.2.4 Populating an Enterprise Directory with Active Directory Server

11.2.5 Populating an H.350 Directory with Active Directory Server

12 Recommended Policies and Procedures for Directory Service Administration

12.1 Account administration

12.2 Protecting the data on the server : Using of ACIs

12.3 Recommended Attribute Indexing

Attribute indexing is an implementation-specific activity. Note that non-indexed attributes can result in search times sufficiently long as to render some applications unusable. Notably, user and alias lookup should be fast. The following Indexing Profile describes an indexing configuration for *commObject* directories that will be optimized for use in directory of directories applications. Use of this profile is optional.

commURI: no recommendation.

commUniqueId: equality

commOwner: presence

commPrivate: presence

h323IdentityGKDomain: no recommendation

h323Identityh323-ID: equality

h323IdentitydialedDigits: equality

h323Identityemail-ID: equality

h323IdentityURL-ID: equality

h323IdentitytransportID: equality

h323IdentitypartyNumber: equality

h323IdentitymobileUIM: equality

h323IdentityEndpointType: equality

h323IdentityServiceLevel: equality

h235IdentityEndpointID: no recommendation

h235IdentityPassword: equality

h320IdentityCC: presence, equality, sub

h320IdentityNDC: presence, equality, sub

h320IdentitySN: presence, equality, sub

h320IdentityExtension: presence, equality, sub

h320IdentityServiceLevel: equality

SIPIdentitySIPURI: equality

SIPIdentityRegistrarAddress: no recommendation

SIPIdentityProxyAddress: no recommendation

SIPIdentityAddress: equality

SIPIdentityUserName: equality

SIPIdentityPassword: no recommendation

SIPIdentityServiceLevel: equality

genericIdentityProtocolIdentifier: equality

genericIdentityMessage: equality

12.4 Securing your server from attacks

13 Registering your Directory Services

13.1 Registering your Directories for GlobalWhite Pages Lookup

13.2 LDAP Crawling and TIO files

14 Using the ViDeNet (Commons? "A"?) Public Directory Service

15 Installing and Configuring Call Servers

15.1 H.323 Gatekeeper Configuration

15.1.1 Installing and Configuring RADVISION ECS

15.1.2 Using ViDeNet GDS

15.2 SIP Proxy Configuration

15.2.1 Installing and Configuring ?????

16 Using Prototype EndPoint (Client) Software

16.1 Installing and Configuring RADVISION Prototype Client

16.2 Installing CGU UserAgent

Obtain the CGU UserAgent at http://ncl.cgu.edu/sipclient/.

16.3 Using RADVISION Prototype

16.4 Using CGU UserAgent Prototype

17 Addressing and Directory Issues for protocol gateways

17.1 Gateways to non-standard protocols

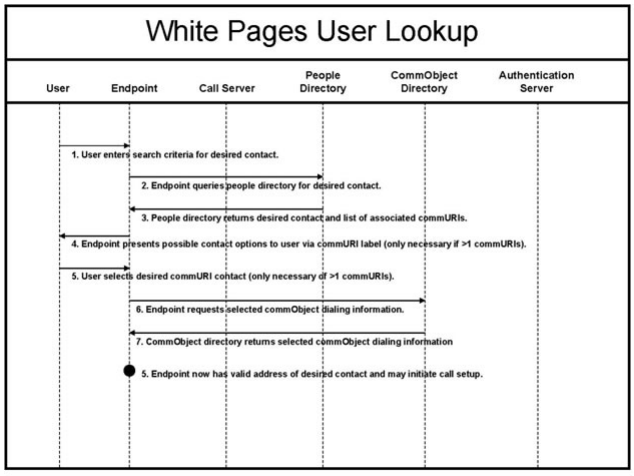
18 Using H.350 for Automated Client Configuration

19 Using H.350 for White Pages Lookup

Figure 19.1 shows how an LDAP-enabled endpoint can search a directory to find a user, obtain that user's *commObject* information, and dial the user's endpoint. In this scenario the endpoint could be pre-configured to search a particular enterprise directory, or it could be preconfigured to search a directory portal (or "Directory of Directories") that it searches many directories. Results of the search can be presented as choices through a user interface. Alternatively, the search criteria could be entered into a web form with results returned and displayed via web page, thus enabling clickable dialing.

The example below illustrates an endpoint doing direct LDAP lookup; it is also possible that a call server could perform lookups on behalf of the endpoint and pass the information to the endpoint through an alternate communication path, thus centralizing some aspects of white page access.

Figure 11: 18.1

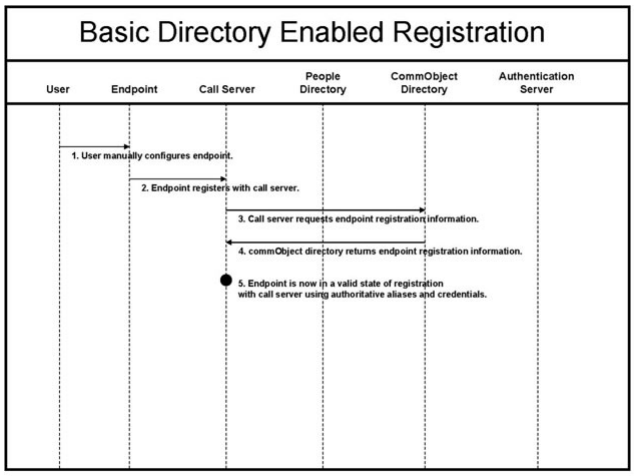


20 Using H.350 as an Authoritative Data Source for Call Servers

20.1 Directory Enabled Registration

Figure 20.1 shows how a call server can access a *commObject* directory to retrieve endpoint information, thus eliminating the need for the call server to have a proprietary internal endpoint database. The call server is always accessing authoritative and up to date information.

Figure 12: 19.1



21 Using H.350 for User Authentication from Authoritative Sources

Figure 21.1 illustrates how an environment can be created in which a user uses an identity authenticated by the enterprise to access any of several endpoint identities. This scenario has several key features that are desirable for large scale deployments.

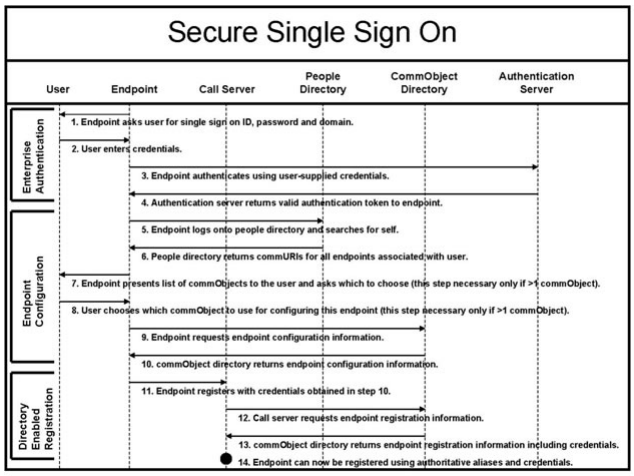
The endpoint is automatically configured using information from the directory. This eliminates user error in the configuration process and simplifies deployment.

The user is needs a single sign on credential (typically their enterprise user ID and password). Endpoints can (and often should) have different credentials, but the user does not need to know them, because they are loaded directly into the endpoint from the directory. Because the endpoint credentials are loaded automatically, it is possible that these credentials could be frequently refreshed. For example, a management tool could generate random credentials for each endpoint and store them in the *commObject* directory each night. This creates

a highly secure environment in which credentials can be very strong, and even if compromised are aged-out and recreated in a short period of time.

This scenario supports the use of ID/password or certificate based endpoint credentials. Certificates have traditionally been found to be difficult to deploy for a number of reasons. This scenario solves some important aspects of the certificate management problem and opens the possibility that certificates can be managed by and on behalf of a central certificate management system, rather than on behalf of users, thus shielding users from the complexity of PKI while gaining its security advantages.

Figure 13: 20.1



In this scenario, the enterprise authentication steps represent a user authenticating to an existing authentication server already deployed for general purpose (e.g. email, web, file sharing) single sign on authentication system. Once authenticated, the user can bind to the LDAP server directly and retrieve all configuration information for the selected endpoint, which includes configuration data and authentication credentials for the endpoint. Finally, using these credentials, the endpoint can authenticate to the call server using whichever authentication scheme is in place (for example, H.235 Annex D or E). Secure LDAP (transport layer security available in LDAP V3) should be used to ensure privacy of these transactions.

Appendix.

22 Example Source Code

22.1 White Pages Lookup

22.1.1 Source Code

As an aid to software developers, some example source code is provided below. The same code can be downloaded here <INSERT LINK TO SOURCE CODE FILE> In this example, the Enterprise and *commObject* directories happen to be hosted on a single ldap server. As explained in the chapters on Architectural Decisions for Implementors, this configuration is just one of several possible architectures. Where an authoritative Enterprise LDAP Directory exists it would be likely that the *commObject* directory would be hosted on a separate directory server.

The scenario here is a white pages lookup - ie, search for someone by name in a person directory; retrieve their commURI's, and present the public information from the eduPerson and *commObject* object classes. Note that this code makes use of anonymous bind, as would be typical for a white pages lookup.

```
/*————————————————————————————————————-
  ldap_example.c
  (1) To Compile on SUN Sparc Station using the SUN
C
      compiler and the openldap libraries (2.1.17 ver-
sion).
      cc -I/opt/local/openldap/include ldap_sample2.c
-o ldap_sample2 -L
      /opt/local/openldap/lib -lldap -llber
  (2) To Compile on RedHat Linux with openldap li-
braries
      cc ./ldap_sample.c -o ldap_sample -lldap -llber
*/

#include <ldap.h>
#include <stdio.h>
/*  NAME OF THE HOST WITH THE LDAP SERVICE
:
  Change this to your server before compiling
*/
#define DEFAULT_HOST "mazurka.dom.unc.edu"
/*  EXAMPLE SEARCH BASE
  Change this before compiling
*/
#define DEFAULT_BASE "uid=tmiller,ou=people,dc=vide,dc=net"
#define PWD "NULL"
#define DEFAULT_FILTER "(objectclass=*)"
```

```
   void print_entries_with_values (LDAP *ld, LDAPMes-
sage *result);
   int url_ldap_search (LDAPURLDesc *);

   int main ()
   {
      LDAP *ld;
      LDAPMessage *result, *e;
      LDAPURLDesc *ludp;
      char **vals;
      char *attrs[2], *url;
      int i, id, err;
      int msgid, nentries, rc;
      int PORT;
      char *SEARCHBASE;
      char *HOST;
      /* get handle to an LDAP connection */
      if ( (ld = ldap_init( DEFAULT_HOST, LDAP_PORT
)) == NULL)
      {
         perror( "ldap_init");
         return (1);
      }
      /* authenticate to the directory anonymously */
     if (ldap_simple_bind_s (ld, NULL, PWD ) != LDAP_SUCCESS)
      {
         ldap_perror (ld, "ldap_simple_bind_s" );
         return (1);
      }
      /* Retrieve commURI */
      attrs[0] = "commuri";
      attrs[1] = NULL;
      if (ldap_search_s(ld, DEFAULT_BASE,
       LDAP_SCOPE_BASE, DEFAULT_FILTER, attrs, 0,
&result) != LDAP_SUCCESS)
      {
         ldap_perror (ld, "ldap_search_s");
         return (1);
      }
      /* print out the values */
      if ( (e = ldap_first_entry(ld, result)) != NULL)
      {
        if ( (vals = ldap_get_values( ld, e, "commuri" )) !=
NULL)
        {
           for ( i = 0; vals[i] != NULL; i++)
```

```
{
  URL = VALS[I];
  PRINTF ("%S\N", VALS[I]);
  /* LET'S PARSE THE URL */
  /* SO WE CAN SEARCH THE SPECIFIED SERVER */
 IF (( ERR = LDAP_URL_PARSE( URL, &LUDP )) != 0 )
  {
       FPRINTF( STDERR, "LDAP_URL_PARSE: ERROR
%D\N", ERR );
  }
  ELSE
  {
    PRINTF( "\T HOST: " );
    IF ( LUDP->LUD_HOST == NULL )
    {
      PRINTF( "DEFAULT\N" );
      HOST = "LOCALHOST";
    }
    ELSE
    {
      PRINTF( "<%S>\N", LUDP->LUD_HOST );
      HOST = LUDP->LUD_HOST;
    }
    PRINTF( "\T PORT: " );
    IF ( LUDP->LUD_PORT == 0 )
    {
      PRINTF( "DEFAULT\N" );
      PORT = LDAP_PORT;
    }
    ELSE
    {
      PRINTF( "%D\N", LUDP->LUD_PORT );
      PORT = LUDP->LUD_PORT;
    }
    SEARCHBASE = LUDP->LUD_DN;
    PRINTF( "\T DN: <%S>\N", LUDP->LUD_DN );
    PRINTF( "\T ATTRS:" );
    IF ( LUDP->LUD_ATTRS == NULL )
    {
      PRINTF( " ALL" );
    }
    ELSE
    {
      FOR ( I = 0; LUDP->LUD_ATTRS[ I ] != NULL;
++I )
      {
```

```
                    printf( " <%s>", ludp->lud_attrs[ i ] );
                  }
                }
                printf( "\n\t scope: %s\n", ludp->lud_scope
== LDAP_SCOPE_ONELEVEL ?
            "ONE" : ludp->lud_scope == LDAP_SCOPE_BASE
? "BASE" :
             ludp->lud_scope == LDAP_SCOPE_SUBTREE
? "SUB" : "**invalid**" );
             printf( "\tfilter: <%s>\n\n", ludp->lud_filter
);
                 /* Let's now call the function that will
follow the LDAP URL and retrieve the information
                */
                rc = url_ldap_search (ludp);
                if (rc == -1 )
                {
                      printf ( " Could not Initialize our
connection to the LDAP server: %s\n",HOST);
                }
                else if (rc == -2)
                {
                     printf ( "Failed to bind to the LDAP
server: %s\n", HOST);
                }
                else if (rc == -3)
                {
                   printf ( "Failed to find any information
for the following filter: %s\n",ludp->lud_filter);
                }
                else if (rc == -4)
                {
                     printf ( "We ran into an error in the
ldap_result function\n");
                }
                ldap_free_urldesc( ludp );
              }
            }
            ldap_value_free (vals);
          }
      }
      ldap_msgfree(result);
      ldap_unbind(ld);
      return (0);
   }
```

```
   /* let's follow up the URL and get the necessary in-
formation by doing
   an LDAP search */
   int url_ldap_search (LDAPURLDesc *ludp)
   {
      int msgid, nentries, rc;
      LDAP *ld;
      char *HOST;
      int PORT;
      LDAPMessage *result;
      /* We need to first initialize the connection to the
LDAP server */
      if ( ludp->lud_host == NULL )
      {
         HOST = "localhost";
      }
      else
      {
         HOST = ludp->lud_host;
      }
      if ( ludp->lud_port == 0 )
      {
         PORT = LDAP_PORT;
      }
      else
      {
         PORT = ludp->lud_port;
      }
      if ( (ld = ldap_init (HOST, PORT)) == NULL)
      {
         perror ( "ldap_init" );
         return (-1);
      }
      /* Now Let's Bind to the server anonymously in this
case */
     /* if (ldap_bind(ld, NULL, NULL, LDAP_AUTH_SIMPLE)
!= LDAP_SUCCESS) */
      if (ldap_simple_bind_s (ld, NULL, NULL) != LDAP_SUCCESS)
      {
         ldap_perror (ld ,"ldap_bind");
         return (-2);
      }
      /* Now Let's call the ldap_search function with the
specific information */
      /* we got from the previous search */
```

```
      if ( (msgid = ldap_search( ld, ludp->lud_dn, ludp-
>lud_scope,
      ludp->lud_filter, NULL, 0 )) == -1)
      {
        ldap_perror( ld, "ldap_search" );
        return (-3);
      }
      /* Let's now list the information we got from the
above search */
      nentries = 0;
      while ( (rc = ldap_result( ld, msgid, 0, NULL, &re-
sult )) == LDAP_RES_SEARCH_ENTRY )
      {
        nentries++;
        print_entries_with_values( ld, result );
        ldap_msgfree( result );
      }
      if ( rc == -1 )
      {
        ldap_perror( ld, "ldap_result" );
        return (-4);
      }
     printf( "Found %d entries at %s\n\n",nentries, ludp-
>lud_dn);
      ldap_unbind( ld );
      return( 0 );
   }

   void print_entries_with_values (LDAP *ld, LDAPMes-
sage *result) {
      LDAPMessage *e;
      BerElement *ber;
      char *dn, *attr;
      char **vals;
      int i;
      for ( e = ldap_first_entry( ld, result); e != NULL;
      e = ldap_next_entry(ld, e) )
      {
        if ( (dn = ldap_get_dn( ld, e)) != NULL)
        {
          printf( "dn: %s\n", dn);
          ldap_memfree( dn );
        }
        for ( attr = ldap_first_attribute( ld, e, &ber );
        attr != NULL;
        attr = ldap_next_attribute( ld, e, ber) )
```

```
        {
            IF ( (VALS = LDAP_GET_VALUES( LD, E, ATTR)) !=
NULL)
          {
            FOR (I = 0; VALS[I] != NULL; I++)
            {
              PRINTF( "%S: %S\N", ATTR, VALS[I] );
            }
            LDAP_VALUE_FREE( VALS);
          }
          LDAP_MEMFREE ( ATTR );
        }
        PRINTF( "\N" );
        IF ( BER != NULL )
        {
          BER_FREE ( BER, 0 );
        }
      }
    }
```

22.1.2 Program Output

The program above has the following output when run successfully:

```
ldap://mazurka.dom.unc.edu/dc=vide,dc=net??sub?(commUniqueId=30)
host: <mazurka.dom.unc.edu>
port: 389
dn: <dc=vide,dc=net>
attrs: ALL
scope: SUB
filter: <(commUniqueId=30)>
dn: commUniqueId=30,ou=h323identity,dc=vide,dc=net
objectClass: top
objectClass: commObject
objectClass: h323Identity
commUniqueId: 30
h323IdentityEndpointType: Terminal
commOwner: ldap://mazurka.dom.unc.edu/dc=vide,dc=net??sub?(uid=tmiller)
h323IdentityDialedDigits: 00121971297
h323Identityh323Id: Theresa.Miller
Found 1 entries at dc=vide,dc=net
ldap://mazurka.dom.unc.edu/dc=vide,dc=net??sub?(commUniqueId=100000)
host: <mazurka.dom.unc.edu>
port: 389
dn: <dc=vide,dc=net>
attrs: ALL
scope: SUB
```

filter: <(commUniqueId=100000)>
dn: commUniqueId=100000,ou=h323identity,dc=vide,dc=net
objectClass: top
objectClass: commObject
objectClass: h323Identity
objectClass: SIPIdentity
commUniqueId: 100000
commOwner: ldap://mazurka.dom.unc.edu/dc=vide,dc=net??sub?(uid=tmiller)
SIPIdentitySIPURI: TheresaMiller@unc.edu
Found 1 entries at dc=vide,dc=net

23 Glossary

ACI (Access Control Item):

ACL (Access Control List):

BIND:

Call Server: a protocol-specific signaling engine that routes video or voice calls on the network. In H.323 this entity is a gatekeeper. In SIP, this entity is a SIP Proxy Server. Note that not all signaling protocols use a call server.

Client: a SIP client is a network device that initiates SIP requests and receives SIP responses on a network.

CN: Common Name

commObject: An LDAP object class defined in ITU-T H.350 that represents generic multimedia conferencing endpoints.

commObject Directory:

commURI:

Directory Server:

DN: Distinguished Name

Endpoint: a logical device that provides video and/or voice media encoding/decoding, and signaling functions. Examples include:

1. a group teleconferencing appliance that is located in a conference room
2. an IP telephone.
3. a software program that takes video and voice from a camera and microphone and encodes it and applies signaling using a host computer.

Enterprise Directory: A canonical collection of information about users in an organization. Typically this information is collected from a variety of organizational units to create a whole. For example, Human Resources may provide name and address, Telecommunications may provide the telephone number, Information Technology may provide the email address, etc. For the purposes of this architecture, it is assumed that an enterprise directory is accessible via LDAP.

Gatekeeper:

Gateway: A device that translates from one protocol to another. Often gateways translate between the IP network and the public switched voice network to allow integration of the two.

Global Dialing Scheme:

H.323:

h323Identity:

H.350:

Internet2: A consortium led by 200 universities working in partnership with industry and government to develop and deploy advanced network applications and technologies, accelerating the creation of tomorrow's Internet. See http://www.internet2.edu/.

LDAP: Lightweight Directory Access Protocol as defined in IETF RFC 1777.

LDIF (LDAP Data Interchange Format):

MCU: Multipoint Control Unit. A device capable of mixing audio/video from multiple endpoints to create a virtual meeting space.

Middleware:

Proxy Server (SIP Proxy): a server that acts as both a client and a server to make requests on behalf of another user agent. The primary role of a proxy server is to ensure that a request generated by a UA is passed to another entity that is closer to the destination user.

RDN: Relative Distinguished Name

Registrar: a registrar is a server that accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles.

Resource: A non-human entity to which an endpoint is associated. For example, and endpoint may be associated with a conference room, classroom, office, or other physical or virtual location.

SIP: Session Initiation Protocol as defined in IETF RFC 3261

SIP URI: a type of Uniform Resource Identifier that identifies a communication resource in SIP. A SIP URI usually contains a user name and a host name and is similar in format to an email address.

User Agent (UA): a device that can function as both a user agent client and server in SIP.

ViDe: The Video Development Initiative. Founded by representatives from universities and education networks, the Video Development Initiative (ViDe) promotes the deployment of digital video in research and higher education. Leveraging collective resources and expertise, ViDe advances digital video deployment through promotion and development of interoperable, standardized, and cost-effective technologies. See http://www.vide.net/.

ViDeNet: ViDeNet is a project of ViDe that consists of a large scale, multi-institutional test bed of interconnected voice and video over IP networks in order to explore issues associated with global deployment of those technologies. See https://videnet.unc.edu/.

VidMid: The Video Middleware Working Group. VidMid is a joint working group sponsored by Internet2 and ViDe. See http://middleware.internet2.edu/video.

White Pages: An application that allows end users to look up the address of another user. This may be web-based or use some other user interface

24 Resources

24.1 H.323 Resources

1. ViDe VideoConferencing Cookbook: Excellent introduction to videoconferencing, written especially for the higher education and K-12 community

2. Paul Jones' H.323 Site at Packetizer.com: Good source for current drafts of H.323 Recommendations and lots of other good information; (Paul is ITU-T Study Group 16 H.323 Rapporteur)

3. OpenH323 Project: Open source implementation of the H.323 protocol

4. h.323 Forum: An industry forum sponsored by sponsored by the International Multimedia Telecommunications Consortium (IMTC). Presentations, Whitepapers and certification information

5. International Engineering Consortium tutorial on H.323: Explains H.323 call flows with an emphasis on gateways and gatekeepers.

6. Wainhouse Whitepapers on rich media communication: "Business case" type approach to evaluating multimedia services

7. Real Time Protocol (RTP): IETF RFC 1889
8. ITU-T Recommendation H.323 (2000), Packet-based multimedia communications systems.
9. ITU-T Recommendation H.235 (2000), Security and encryption for H-Series (H.323 and other H.245-based) multimedia terminals.
10. ITU-T Recommendation H.225.0 (2000), Call signaling protocols and media stream packetization for packet-based multimedia communications systems.
11. ITU-T Recommendation H.320 (1999), Narrow-band visual telephone systems and terminal equipment.

24.2 SIP Resources

1. IETF SIP Working Group
2. Columbia University SIP site
3. SIP Center: commercial developers' site; whitepapers & other information
4. SIP Forum: non-profit organization of SIP developers
5. SIPdev: useful information site for SIP developers
6. IETF RFC 2617 (1999), HTTP Authentication: Basic and Digest Access Authentication.
7. IETF RFC 3261 (2002), SIP: Session Initiation Protocol.
8. IETF RFC 3263 (2002), Session Initiation Protocol (SIP): Locating SIP Servers.

24.3 LDAP Resources

1. IETF RFC 3377 Lightweight Directory Access Protocol (v3) Technical Specification
2. A Recipe for Configuring and Operating LDAP Directories by Michael R. Gettes.
3. LDAP Roadmap and FAQ
4. IETF RFC 2252 Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions
5. Gerald Carter (2003), O'Reilly and Associates, ISBN: 1565924916, LDAP System Administration.

6. Timothy A. Howes, PhD, Mark C. Smith, Gordon S. Good, New Riders Publishing (1999), ISBN: 1578700701, Understanding And Deploying LDAP Directory Services.

7. Timothy A. Howes, PhD, Mark C. Smith, New Riders Publishing (1997), ISBN: 1578700000, LDAP Programming Directory-Enabled Applications with Lightweight Directory Access Protocol.

8. iPlanet Directory Server 5.1 Documentation

9. iPlanet Directory Server Kit 5.1 Tools Reference

10. SunOne Directory Server 5.2 (Multi-Platform) Documentation

24.4 Middleware Resources

24.5 Directory of Directories Resources

1. Description of functional design of the Desire LDAP/TIO-Index server ((Desire was a European project on Development of a European Service for Information on Research and Education)