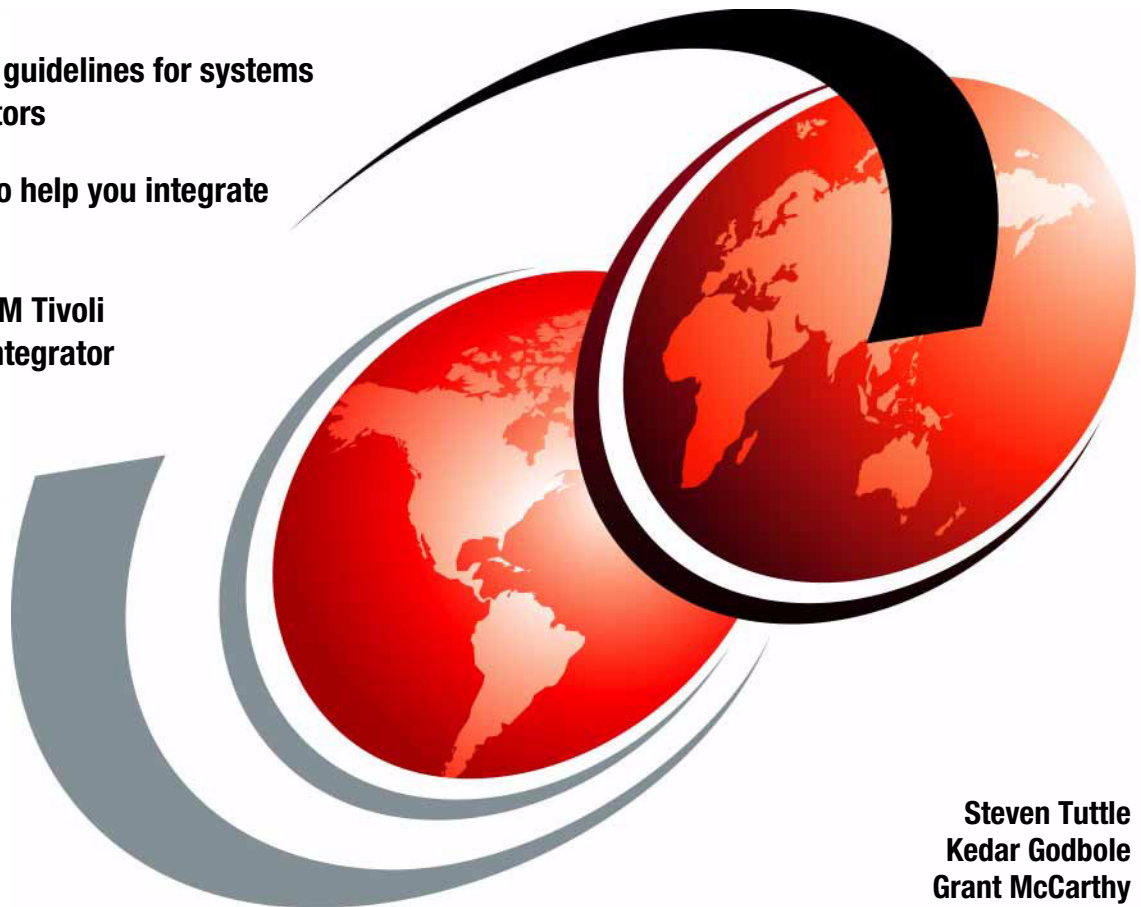**IBM**

Using LDAP for Directory Integration

Integration guidelines for systems administrators

Examples to help you integrate directories

Includes IBM Tivoli Directory Integrator

Steven Tuttle
Kedar Godbole
Grant McCarthy

Redbooks

International Technical Support Organization

Using LDAP for Directory Integration

February 2004

Note: Before using this information and the product it supports, read the information in "Notices" on page vii.

Second Edition (February 2004)

This edition applies to IBM Tivoli Directory Server V5 release 2, IBM Tivoli Directory Integrator V5 release 2, IBM Lotus Domino Server V6 release 5, and Microsoft Windows 2000 Advanced Server Active Directory.

Contents

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

@server®	Domino®	Redbooks™
@server®	DB2®	RDN™
Redbooks (logo) ™	IBM®	SecureWay®
Eserver™	Lotus Notes®	Tivoli Enterprise™
ibm.com®	Lotus®	Tivoli®
iSeries™	Notes®	WebSphere®

The following terms are trademarks of other companies:

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, and service names may be trademarks or service marks of others.

Preface

This IBM Redbook is intended for directory administrators who are faced with today's directory management challenges.

This book explains directory and LDAP concepts, and it deals with some basic and advanced scenarios that demonstrate ways to tackle directory integration challenges. These scenarios use some leading products that are available including the IBM Tivoli Directory Server, IBM Tivoli Directory Integrator, IBM Lotus Domino, and Microsoft's Active Directory.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center.

Steven Tuttle is a Project Leader for the International Technical Support Organization (ITSO), Austin center. He has 13 years of experience in the IT industry. He has worked at IBM for 10 years, with 5 years of experience with IBM security products. He holds a degree in Computer Science from Clarkson University in Potsdam, New York, with concentrations in Mathematics and Psychology. His areas of expertise include the IBM Tivoli Enterprise products and the IBM Tivoli Security products. Before joining the ITSO, he worked for IBM Tivoli Services in the Security Practice as an enterprise security solution designer using IBM Tivoli software products.

Kedar Godbole is a Software Engineer working with IBM Global Services in Pune, India. He has over four years of experience in the IT industry. He has worked for IBM for over two years. He holds a master's degree in Electrical Engineering from Government College of Engineering in Pune, India with a specialization in Control Systems. His areas of expertise include products like IBM Tivoli Directory Integrator and IBM Transarc Distributed File System.

Grant McCarthy is an Advisory IT Specialist with IBM South Africa. He is certified in Domino Administration and has 7 years of administration and development experience in the Domino/Notes environment. He has recently received the EMEA 2003 Outstanding Technical Achievement Award for his work on an Domino/Notes project in the security environment for South Africa. He has also developed numerous other Domino applications that are being used by IBM South Africa. Currently, he is the Domino Administrator for a large account in IBM Global Services, South Africa.

Thanks to the following people for their contributions to this project:

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners, and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

> `http://www.ibm.com/redbooks/residencies.html`

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

► Use the online **Contact us** review redbook form found at:

  `http://www/ibm.com/redbooks`

► Send your comments in an Internet note to:

  `mailto:redbook@us.ibm.com`

► Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. JN9B Building 003 Internal Zip 2834
  11400 Burnet Road
  Austin, Texas 78758-3493

Summary of changes

This section describes the technical changes made in this edition of the book and in previous editions. This edition may also include minor corrections and editorial changes that are not identified.

Summary of Changes
for SG24-6163-01
for Using LDAP for Directory Integration
as created or updated on February 11, 2004.

February 2004, Second Edition

This revision reflects the addition, deletion, or modification of new and changed information described below.

New information
► Overview and Use of IBM Tivoli Directory Integrator

Changed information
► Updated the versions of products used in the scenarios
► Scenarios updated or created to use the functionality of ADSync and IBM Tivoli Directory Integrator when possible.

1

Introduction to Directory Integration

This chapter introduces the concept and advantages of directories in the corporate environment. It describes the Lightweight Directory Access Protocol (LDAP) and defines standards and terminologies involved in accessing directories. This provides a framework for discussing common directory definitions, the particularities of implementation, and the part that a corporate directory can play in an integrated environment. Finally, it introduces the applications that make use of LDAP, which are used in this book.

1.1 Introduction

People and businesses are increasingly relying on networked computer systems to support distributed applications. These distributed applications might interact with computers on the same local area network (LAN), within a corporate intranet, or anywhere in the world on the Internet. To improve functionality, provide ease of use, and enable cost-effective administration of distributed applications, information about the services, resources, users, and other objects accessible from the applications needs to be organized in a clear and consistent manner. Much of this information can be shared among many applications, but it must also be protected to prevent unauthorized modification or the disclosure of private information. Security, however, is not the only consideration when applying a service policy to a piece of communication. The quality of service to be delivered is another major element, and directories today are capable of holding millions of objects.

Information describing the various users, applications, files, printers, and other resources accessible from a network is often collected into a special database that is sometimes called a directory. As the number of different networks and applications has grown, the number of specialized directories of information has also grown. This growth results in islands of information that are difficult to share and manage. If all of this information could be maintained and accessed in a consistent and controlled manner, it would provide a focal point for integrating a distributed environment into a consistent and seamless system.

LDAP is an open industry standard that has evolved to meet these needs. LDAP defines a standard method for accessing and updating information in a directory. LDAP is gaining wide acceptance as the directory access method of the Internet and is therefore also becoming strategic within corporate intranets. It is being supported by a growing number of software vendors and is being increasingly incorporated into applications. For example, the two most popular Web browsers, Netscape Navigator/Communicator and Microsoft Internet Explorer, support LDAP as a base feature.

1.2 Directories

A directory is a listing of information about objects arranged in some order that gives details about each object. Common examples are a city telephone directory and a library card catalog. For a telephone directory, the objects listed are people; the names are arranged alphabetically, and the details given about each person are address and telephone number. Books in a library card catalog are ordered by author or by title, and information such as the ISBN number of the book and other publication information is given.

In computer terms, a directory is a specialized database, also called a data repository, that stores typed and ordered information about objects. A particular directory might list information about printers (the objects) consisting of typed information such as location (a formatted character string), speed in pages per minute (numeric), print streams supported (for example PostScript or ASCII), and so on.

Directories allow users or applications to find resources that have the characteristics needed for a particular task. For example, a directory of users can be used to look up a person's e-mail address or fax number. A directory can be searched to find a nearby PostScript color printer. Finally, a directory of application servers could be searched to find a server that can access customer billing information.

The terms *white pages* and *yellow pages* are sometimes used to describe how a directory is used. If the name of an object (such as a person or printer) is known, its characteristics (such as phone number or pages per minute) can be retrieved. This is similar to looking up a name in the white pages of a telephone directory. If the name of a particular individual object is not known, the directory can be searched for a list of objects that meet a certain requirement. This is like looking up a listing of hairdressers in the yellow pages of a telephone directory. However, directories stored on a computer are much more flexible than the yellow pages of a telephone directory because they can usually be searched by specific criteria, not just by a predefined set of categories.

A directory is often described as a database, but it is a specialized database that has characteristics that set it apart from general-purpose relational databases. One special characteristic of directories is that they are accessed (read or searched) much more often than they are updated (written). Just as hundreds of people might look up an individual's phone number, thousands of print clients might look up the characteristics of a particular printer. However, the phone number or printer characteristics rarely change.

Because directories must be able to support high volumes of read requests, they are typically optimized for read access. Write access might be limited to system administrators or to the owner of each piece of information. A general-purpose database, on the other hand, needs to support applications, such as airline reservations and banking applications, with relatively high-update volumes.

Because directories are meant to store relatively static information and are optimized for that purpose, they are not appropriate for storing information that changes rapidly. For example, the number of jobs currently in a print queue probably should not be stored in the directory entry for a printer because that information would have to be updated frequently to be accurate. Instead, the directory entry for the printer can contain the network address of a print server. The print server can be queried to get the current queue length if desired. The

information in the directory (the print server address) is static, while the number of jobs in the print queue is dynamic.

Another difference between directories and general-purpose databases is that most directory implementations still do not support transactions. However, transactions are supported in LDAP. These transactions are limited to transactions within the LDAP directory and do not include other transactions (for example, database operations). Transactions are all-or-nothing operations that must be completed in total or not at all. For example, when transferring money from one bank account to another, the money must be debited from one account and credited to the other account in a single transaction. If only half of this transaction completes or if someone accesses the accounts while the money is in transit, the accounts will not balance. General-purpose databases usually support such transactions, which complicates their implementation.

Because general-purpose databases must support arbitrary applications such as banking and inventory control, they allow arbitrary collections of data to be stored. Directories may be limited in the type of data they allow to be stored (although the architecture does not impose such a limitation). For example, a directory specialized for customer contact information might be limited to storing only personal information such as names, addresses, and phone numbers. If a directory is extensible, it can be configured to store a variety of types of information, making it more useful to a variety of programs.

Another important difference between a directory and a general-purpose database is in the way information can be accessed. Most databases support a standardized and very powerful access method called Structured Query Language (SQL). SQL allows complex update and query functions at the cost of program size and application complexity. Directories, such as an LDAP directory, on the other hand, use a simplified and optimized access protocol that can be used in slim and relatively simple applications.

Because directories are not intended to provide as many functions as general-purpose databases, they can be optimized to economically provide more applications with rapid access to directory data in large distributed environments. If your intended use of the directory is to be read, mostly in a non-transactional environment, both the directory client and directory server can be simplified and optimized.

A request is typically performed by the directory client, and the process that looks up information in the directory is called the directory server. In general, servers provide a specific service to clients. Sometimes a server might become the client of other servers in order to gather the information necessary to process a request.

A directory service is only one type of service that might be available in a client/server environment. Other common examples of services are file services, mail services, print services, Web page services, and so on. The client and server processes might or might not be on the same machine. A server is capable of serving many clients. Some servers can process client requests in parallel. Other servers queue incoming client requests for serial processing if they are currently busy processing another client's request.

An application programming interface (API) defines the programming interface a particular programming language uses to access a service. The format and contents of the messages exchanged between client and server must adhere to an agreed-upon protocol.

LDAP defines a message protocol used by directory clients and directory servers. The LDAP Protocol uses a variety of messages. For example, a bindRequest may be sent from the client to the LDAP server at the beginning of a connection. A searchRequest is used to search for a specific entry in the directory.

There are also associated LDAP APIs for the C language and ways to access LDAP from within a Java application. Additionally, within the Microsoft development environment, you can access LDAP directories through its Active Directory Service Interface (ADSI). In general with LDAP, the client is not dependent upon a particular implementation of the server; the server can implement the directory however it chooses.

LDAP defines a communication protocol. That is, it defines the transport and format of messages used by a client to access data in an X.500-like directory. LDAP does not define the directory service itself. When people talk about the LDAP directory, they refer to the information that is stored and can be retrieved by the LDAP protocol.

The Comite Consultatif International Telephonique et Telegraphique or Consultative Committee on International Telephony and Telegraphy (CCITT), which is now called the International Telecommunications Union - Telecommunication Standardization Sector (ITU-T), defined the X.500 standard in 1988. The X.500 standard then became International Standards Organization (ISO) 9594, Data Communications Network Directory, Recommendations X.500/X.521 in 1990, though it is still commonly referred to as X.500.

X.500 organizes directory entries in a hierarchical name space capable of supporting large amounts of information. It also defines powerful search capabilities to make information retrieval easier. Because of its functionality and scalability, X.500 is often used together with add-on modules for interoperation between incompatible directory services.

X.500 specifies that communication between the directory client and the directory server uses the Directory Access Protocol (DAP). However, as an application layer protocol, the DAP requires the entire Open Systems Interconnection (OSI) protocol stack to operate. Supporting the OSI stack requires more resources than are available in many small environments. Therefore, an interface to an X.500 directory server using a less resource-intensive or lightweight protocol (in this case, LDAP) is desired.

An application client program initiates an LDAP message by calling an LDAP API. But an X.500 directory server does not understand LDAP messages. In fact, the LDAP client and X.500 server even use different communication protocols (TCP/IP versus OSI). The LDAP client actually communicates with a gateway process (also called a proxy or front end) that forwards requests to the X.500 directory server. This gateway is known as an LDAP server. It services requests from the LDAP client. It does this by becoming a client of the X.500 server. At the beginning, the LDAP server implementations supported both OSI and TCP/IP to be able to translate requests received by LDAP clients to DAP requests required to access X.500 directories. Newer LDAP server implementations, such as the IBM SecureWay Directory server, support only the LDAP protocol to access the directory. The LDAP server on the iSeries server is called Directory Services and implements the IBM SecureWay Directory.

All modern LDAP directory servers are based on LDAP Version 3. You can use a Version 2 client with a Version 3 server. However, you cannot use a Version 3 client with a Version 2 server unless you bind as a Version 2 client and use only Version 2 APIs.

All LDAP servers share many basic characteristics since they are based on industry-standard Request for Comments (RFCs). However, due to implementation differences, they are not all completely compatible with each other when there is not a standard defined.

1.3 Advantages of using a directory

An application-specific directory stores only the information needed by a particular application and is not accessible by other applications. Because a full-function directory service is complex to build, application-specific directories are typically very limited. They probably store only a specific type of information, do not have general search capabilities, do not support replication and partitioning, and probably do not have a full set of administration tools. An application-specific directory could be as simple as a set of editable text files, or it could be stored and accessed in an undocumented, proprietary manner.

In such an environment, each application creates and manages its own application-specific directory, which quickly becomes an administrative nightmare. The same e-mail address stored by the calendar application might also be stored by a mail application and by an application that notifies system operators of equipment problems. Keeping multiple copies of information up-to-date and synchronized is difficult, especially when different user interfaces and even different system administrators are involved.

What is needed is a common, application-independent directory. If application developers could be assured of the existence of a directory service, then application-specific directories would not be necessary. However, a common directory must address the problems mentioned above. It must be based on an open standard that is supported by many vendors on many platforms. It must be accessible through a standard API. It must be extensible so that it can hold the types of data needed by arbitrary applications. Also, it must provide full functionality without requiring excessive resources on smaller systems. Since more users and applications will access and depend on the common directory, it must also be robust, secure, and scalable.

When such a directory infrastructure is in place, application developers can devote their time to developing applications instead of application-specific directories. In the same way that developers rely on the communications infrastructure of TCP/IP and remote procedure call (RPC) to free them from low-level communication issues, they must be able to rely on powerful, full-function directory services. LDAP is the protocol to be used to access this common directory infrastructure. Like HTTP (hypertext transfer protocol) and FTP (file transfer protocol), LDAP has become an indispensable part of the Internet's protocol suite.

When applications access a standard common directory that is designed in a proper way instead of using application-specific directories, redundant and costly administration can be eliminated and security risks are more controllable. For example, the telephone directory, mail, and Web applications shown below can all access the same directory to retrieve an e-mail address or other information stored in a single directory entry. The advantage is that the data is kept and maintained in one place. Various applications can use individual attributes of an entry for different purposes (assuming that the they have the correct authority). New uses for directory information will be realized, and a synergy will develop as more applications take advantage of the common directory. Figure 1-1 on page 8 depicts the advantages of this arrangement.
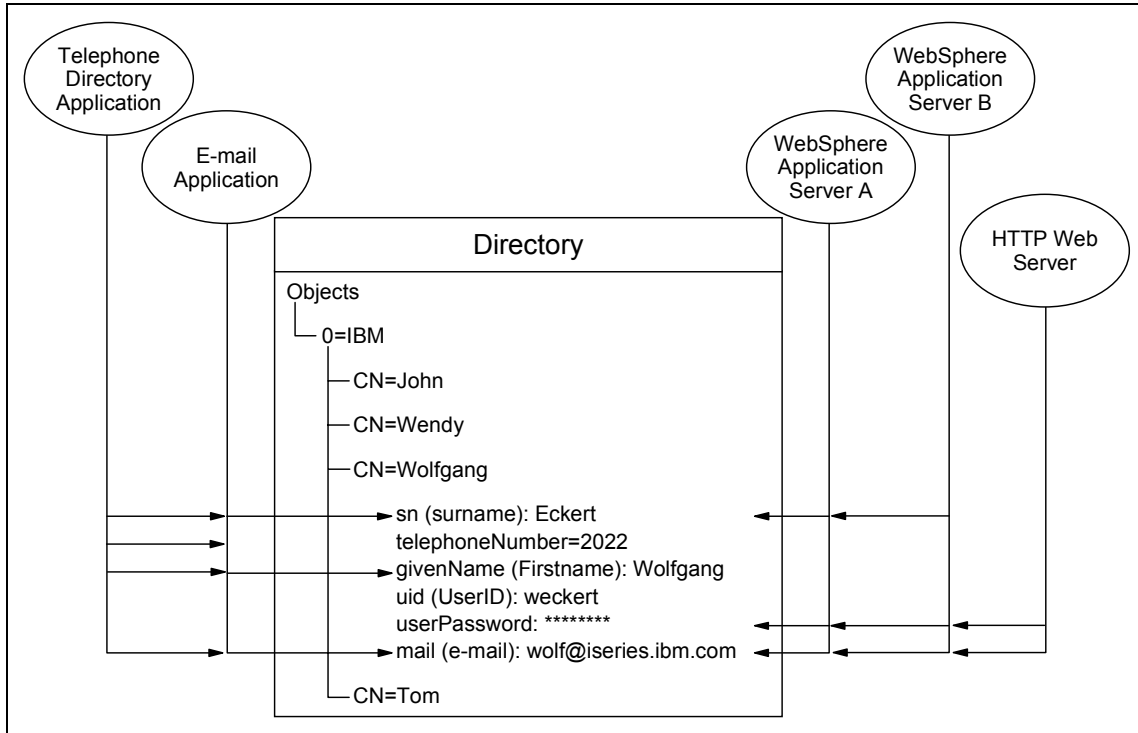
Figure 1-1   Several Applications using attributes of the same entry

Storing data in a directory and sharing it among applications saves you time and money by keeping administration effort and system resources down. Many IBM applications also utilize directories to centrally store and share information. The number of applications that support LDAP directories is constantly increasing. For example, LDAP directory support, such as for authentication and configuration management, is provided in various IBM Operating Systems, IBM WebSphere Application Server, WebSphere Portal Server, Tivoli Access Manager, Directory Server (Directory Server), IBM HTTP server, Lotus Domino, and so on.

1.4 Directory Components

A directory contains a collection of objects organized in a tree structure. The LDAP naming model defines how entries are identified and organized. Entries are organized in a tree-like structure called the Directory Information Tree (DIT). Entries are arranged within the DIT based on their distinguished name (DN). A DN is a unique name that unambiguously identifies a single entry. DNs are made

up of a sequence of relative distinguished names (RDNs). Each RDN in a DN corresponds to a branch in the DIT leading from the root of the DIT to the directory entry. A DN is composed of a sequence of RDNs separated by commas, such as `cn=thomas,ou=itso,o=ibm`.

You can identify common names (CNs) within DNs. You also can organize entries, for example, after organizations. You can further split the tree into organizational units within a single organization as needed. You can define your DIT based on your organizational needs as in the simple example below. If you have, for example, one company with different divisions, you may want to start with your company name under the root as the organization (o) and then branch into organizational units (ou) for the individual divisions. In case you store data for multiple organizations within a country, you may want to start with a country (c) and then branch into organizations. Figure 1-2 provides an example of this approach.

Figure 1-2   Example of a directory information tree
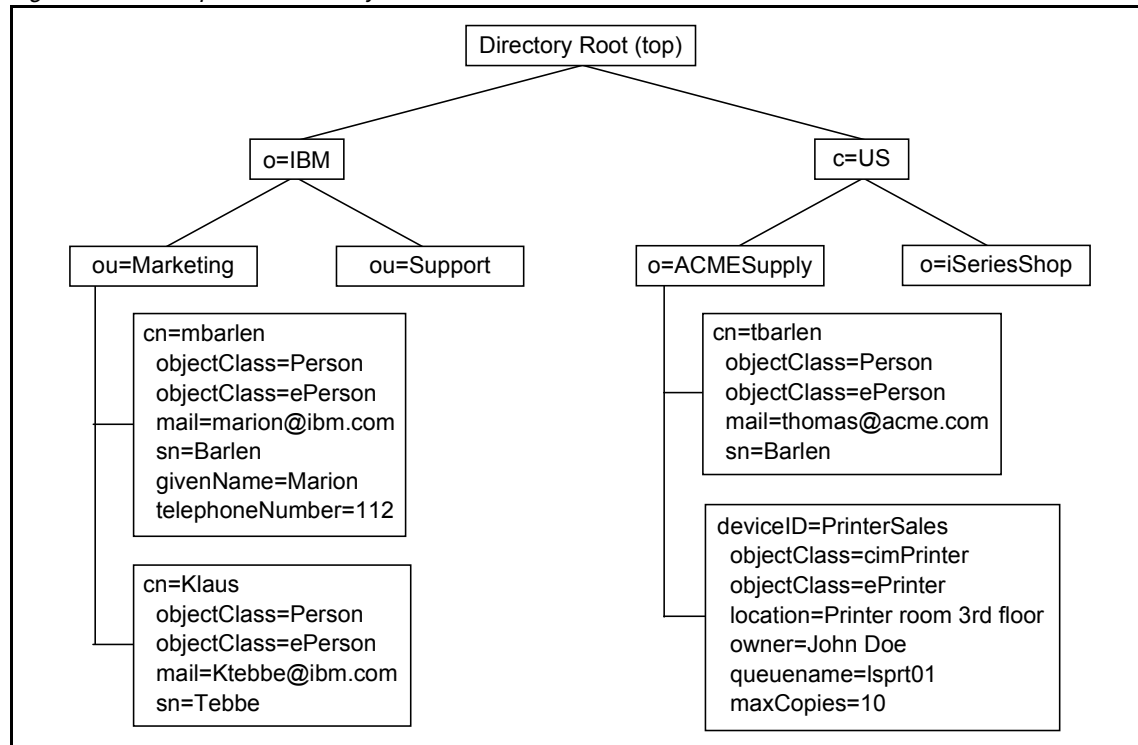
Each object also referred to as an entry in a directory belongs to one or more object classes. An object class describes the content and purpose of the object. It also contains a list of attributes, such as a telephone number or surname, that can be defined in an object of that class. You can publish entries of different

object classes under another object. This is shown in Figure 1-2 on page 9, where an ePrinter object and a Person object are published under the organization ACMESupply.
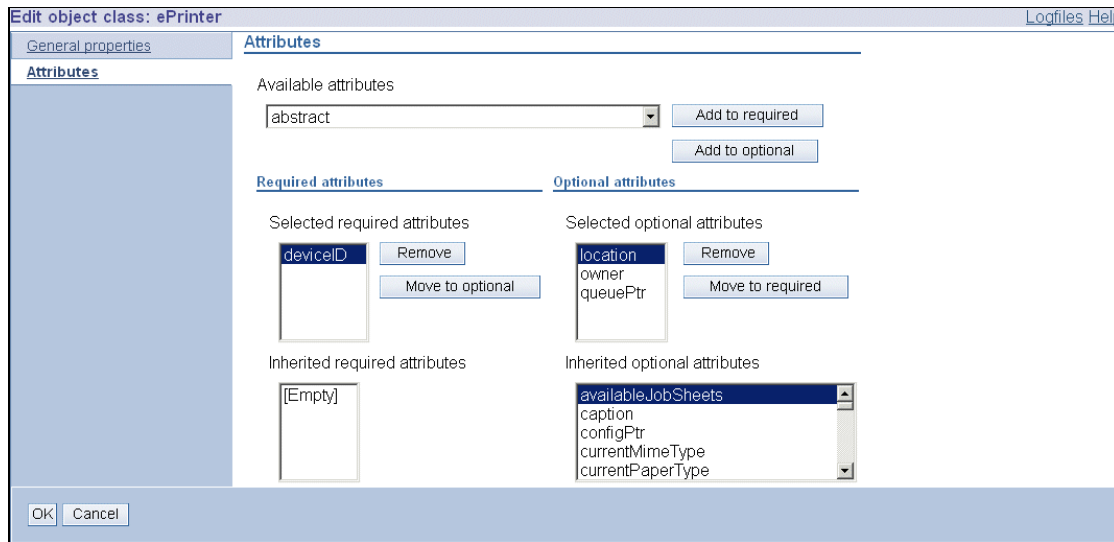

Figure 1-3   ePrinter Object Class

The object class also defines which of the attributes must be defined (are required) when creating an object of this class and which attributes are optional. As shown in Figure 1-3, the object class with the name ePrinter has a required attribute deviceID and three optional attributes that may or may not be filled in when creating an ePrinter object. Object classes also can inherit characteristics, such as attributes from other object classes. In the example of the ePrinter, the class inherits all of the attributes that are defined in the class cimPrinter. Therefore, you must define the deviceID when you create an ePrinter object. Optionally, you can specify the location, owner, and queuePtr attribute of ePerson and all of the attributes of cimPrinter.

Attributes themselves also have certain characteristics. The surname attribute name, for example, is defined as sn and surName and describes a person's family name. The attribute definition also specifies the syntax rules for the attribute value. A telephone number may only contain numbers and hyphens, while the surname consists of alpha characters. Other specifications include whether this attribute can contain only one or many values, the matching rules, the Object Identifier (OID), and so on. The Directory Server (ITDS) product also includes some IBM proprietary extensions for each attribute. Other manufacturers, such as Microsoft, have similar extensions. The IBM extensions include also an access class, which is used in combination with access control

lists (ACLs) to control who can perform a certain action on the attribute value (such as read, write, search, or compare operations).

All the objects and attributes with their characteristics are defined in schemas. The schema specifies what can be stored in the directory. Schema-checking ensures that all required attributes for an entry are present before an entry is stored. Schema-checking also ensures that attributes not in the schema are not stored in the entry. Optional attributes can be filled in at any time. A schema also defines the following:

► Inheritance
► Subclassing of objects
► Where in the DIT structure (hierarchy) objects may appear

Information about the IBM Tivoli Directory Schema schema can be found at:

    http://publib.boulder.ibm.com/tividd/td/IBMDS/IDSschema52/en_US/HTML/schema
    .html

An attribute definition is shown in Figure 1-4.

Figure 1-4   Attribute definition example

Object classes and attributes including their specifications are defined as OIDs in an ASN.1 notation format. All these OIDs are registered with a public organization, such as the American National Standards Institute (ANSI) organization (http://www.ansi.org) for the United States. The number notation refers to a hierarchy. For example, the OID 2.5.4.4 resolves into a surName attribute as shown in Figure 1-5.
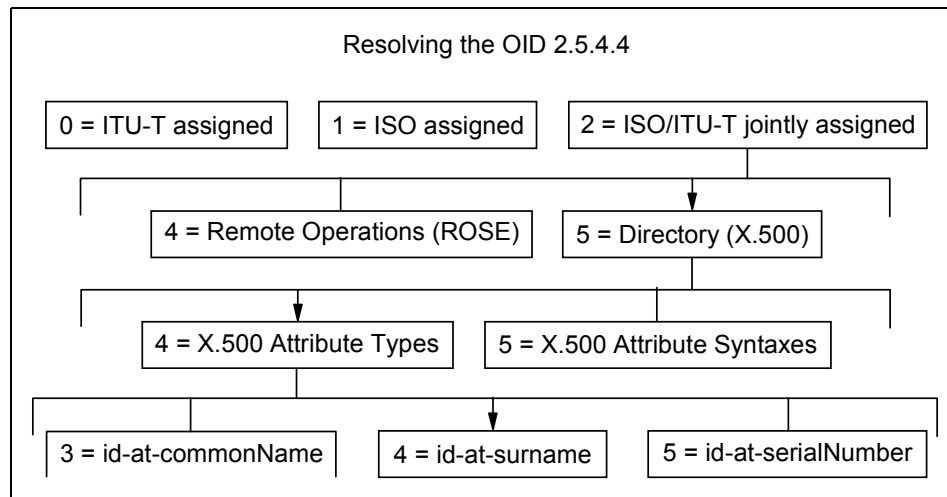
.

Figure 1-5   Example of Object Identifiers as defined by the ANSI organization

1.5 IBM Tivoli Directory Integrator

IBM Tivoli Directory Integrator (Directory Integrator) synchronizes identity data residing in directories, databases, collaborative systems, applications used for human resources (HR), customer relationship management (CRM), and Enterprise Resource Planning (ERP), and other corporate applications.

By serving as a flexible, synchronization layer between a company's identity structure and the application sources of identity data, Directory Integrator eliminates the need for a centralized datastore. For those enterprises who do choose to deploy an enterprise directory solution, Directory Integrator can help ease the process by connecting to the identity data from the various repositories throughout the organization.

With some built-in connectors, an open-architecture Java development environment to extend or modify these connectors, and tools to apply logic to data as

data is processed, Directory Integrator can help you by:

► Synchronizing and exchanging information between applications or directory sources

► Managing data across a variety of repositories, thus providing the consistent directory infrastructure needed for a wide variety of applications that include security and provisioning

► Creating the authoritative data spaces needed to expose only trustworthy data to advanced software applications such as Web services

1.5.1 Directory Integrator software components

The components of the Directory Integrator include the following:

► An Assembly Line methodology that
  – Builds a compound information object from connected information sources
  – Performs modifications on received data
  – Creates new entries altogether and adds/updates/deletes the new information object to the assigned destinations

► An Event Handler framework that adds to the flexibility of Directory Integrator by providing the ability to wait for, and react to, specific events that have taken place in the infrastructure: These events include changes in a directory, arriving e-mails, records updated in certain databases, incoming HTML pages from a Web server or browser, arriving Web services-based Simple Object Access Protocol (SOAP) messages, and other types of events defined by the user.

► Connectors to support numerous protocols and access mechanisms that are included with the product or can be easily created or modified

► Parsers to interpret and translate information from a byte stream into a structure information object, where each piece of information is accessible by name: You also can translate a structured information object into a byte stream. You can select from the wide range of extensible parsers such as *comma separated values*, *fixed column*, LDAP Data Interchange Format (LDIF), Extensible Markup Language (XML), SOAP, and Directory Services Markup Language (DSML). Finally, you can create a new parser from scratch.

► Hooks to enable the definition of certain actions to be executed under specific circumstances or at desired points in the execution of the AssemblyLine process

► Password Synchronization plug-ins that are separately installable entities available with Directory Integrator: These plug-ins can detect when a

password is changed or created. Once the plug-ins detect this change, they act to propagate the information to a backend store.

The plug-and-play functionality of these Directory Integrator components facilitate rapid prototyping and implementation of intelligent data flows. Additionally, it is possible to extend virtually all of these integration components (such as Connectors, Parsers, and EventHandlers), functions, and attributes through Java scripts. Directory Integrator supports JavaScript and Perl plug-in scripting languages that can be used with every AssemblyLine, Parser, and Connector.

1.5.2 Solution building approach

Directory Integrator is designed and built on the premise that integration problems can be broken down into three basic components:

► **Systems** involved in the communication
► **Data flows** between these systems
► **Events** that trigger the data flows

With Directory Integrator, you turn this atomic understanding of the integration problem directly into the solution. You can build your solution incrementally, one flow at a time, with continuous feedback and verification.

This means that integration projects become easier to estimate and plan. Since you are developing the solution flow-by-flow visually and interactively, you can report (and demonstrate) progress to both project and corporate management at any time.

Directory Integrator manages the technicalities of connecting to and interacting with the various data sources that you want to integrate, abstracting away the details of their APIs, transports, protocols and formats. Instead of focusing on data, Directory Integrator lifts your view to the information level, enabling you to concentrate on the transformation, filtering, and other business logic required to perform each exchange.

Directory Integrator enables you to build libraries of components and business logic that can be maintained, extended, and reused to address new challenges. Development projects across your organization can all share Directory Integrator assets, resulting in independent projects (even point solutions) that immediately fit into a coherent and integrated infrastructure.

This approach results in a more rational and predictable use of resources, as you bring your data source and technology experts in at the very start of a project in order to set up your libraries. When in place, these integration assets are

available across the network, letting you leverage them to create new solutions and enhance existing ones.

1.6 IBM Tivoli Directory Server

IBM Tivoli Directory Server (ITDS) is the IBM LDAPv3 Directory offering. ITDS implements the Internet Engineering Task Force (IETF) LDAP V3 specifications. It also includes enhancements added by IBM in functional and performance areas. This version uses IBM DB2 Universal Database as the backing store to provide per LDAP operation transaction integrity, high performance operations, and online backup and restore capability. ITDS interoperates with the IETF LDAP V3-based clients. Major features include:

► A Graphical User Interface (GUI) that can be used to administer and configure the IBM Directory: The administration and configuration functions enable the administrator to

  – Perform the initial setup of the directory

  – Change configuration parameters and options

  – Manage the daily operations of the directory, such as adding or editing objects (for example, object classes, attributes, and entries)

► A dynamically extensible directory schema: This means that administrators can define new attributes and object classes to enhance the directory schema. Changes can be made to the directory schema, also, that are subject to consistency checks. Users may dynamically modify the schema content without restarting the directory server. Because the schema itself is part of the directory, schema update operations are done through standard LDAP APIs. The major functions provided by the LDAPv3 dynamic extensible schema are

  – Queriable schema information through LDAP APIs

  – Dynamic schema changes through LDAP APIs

  – Server Root Directory Services Engine (DSE)

► UTF-8 (Universal Character Set Transformation Format): An Directory Server supports data in multiple languages and allows users to store, retrieve and manage information in a native language code page.

► Simple Authentication and Security Layer (SASL): This support provides for additional authentication mechanisms. The Secure Sockets Layer (SSL) provides encryption of data and authentication using X.509v3 public-key certificates. A server may be configured to run with or without SSL support.

► Replication: Replication is supported, which makes additional read-only copies of the directory available, improving performance and reliability of the

directory service. Replication topologies also support forwarding and gateway servers.

► Referrals: Support for LDAP referrals is provided, allowing directories to be distributed across multiple LDAP servers where a single server may contain only a subset of the whole directory data.

► Access control model: A powerful, easy-to-manage access control model is supported through ACLs.

► Change log

► Password policy

► Security audit logging

► Dynamic configuration changes using LDAP APIs

1.7 IBM Lotus Domino 6.5

IBM Lotus Domino is an enterprise-class messaging and collaboration system, designed to take full advantage of the e-business revolution. It runs on a variety of different hardware platforms and operating systems.

IBM Lotus Domino server supports industry standards like Simple Mail Transfer Protocol (SMTP), Multipurpose Internet Mail Extensions (MIME), Post Office Protocol (POP3), LDAP, and SSL.

IBM Lotus Domino is designed to simplify integration into a multi-directory environment. With IBM Lotus Domino (Domino) 6 (or later), you have the option of moving from a distributed directory architecture and making Domino the central directory. This allows you to take advantage of a centralized directory configuration that provides added control and less overhead and is easier to manage. Domino Server comes with the Domino Upgrade Services tool. This tool is used to import users from a server-based foreign directory and register those users in the Domino Directory. Domino Upgrade Services migrates data from many different systems, some of which include LDAP Data Interchange Format (LDIF) files, LDAP-compliant foreign directories (such as Directory Server), Microsoft Windows NT Server, and Active Directory.

IBM Lotus Domino 6.5 also has enhanced the implementation of LDAP capabilities and improved the performance of LDAP directory access. A new Domino LDAP Schema database allows you to maintain and extend the schema. Other directory schemas can be imported via LDIF files.

Other Domino features include the following:

► Support for X.500 naming conventions, including hierarchical naming and extensible attributes, for maximum flexibility in configuring the namespace

► LDAP protocol support in both the client and the server providing lookup (read), add, delete, and modify (write) support for non-Notes clients (for example Web browsers) and servers such as Active Directory (Active Directory) and Four11: This support allows for developing a distributed user authentication mechanism, making it easier to achieve a single sign-on environment.

► Rule-based domain relationships for faster lookups across large namespaces

► Hierarchical naming and trust between domains to support the relationship of entries across domains

► Support for a Public Key Infrastructure

► A dynamically extensible directory schema, ideal for customizing the directory to meet specific business requirements)

► Multi-master replication, a key element for reliable directory synchronization and maximum availability

► An open architecture that can easily incorporate support for emerging standards

1.8 Microsoft Active Directory

The Microsoft Windows 2000 operating system includes Active Directory, an LDAP Version 3-compliant directory service. Active Directory follows RFC 2247 conventions for naming contexts, meaning that its naming context suffixes map directly to the DNS tree.

Some Active Directory features include:

► Centralized management
► Group policies
► A global catalog
► IntelliMirror desktop management and automated software distribution
► Multi-master replication
► Kerberos authentication
► Smart card support
► PKI/x.509 certificate support
► Attribute-level security

1.8.1 Naming contexts

In Active Directory, naming contexts are the primary unit of replication. Active Directory always has at least three naming contexts:

► The schema

► The configuration, which holds the replication topology and metadata

► User naming contexts, which are the body of the directory and hold the actual directory objects

1.8.2 Logical elements

Logical elements in Active Directory include the following:

1. Domains: Active Directory is made up of at least one domain. Microsoft defines a domain as "a logical grouping of network servers and other computers that share common security and user account information." The main implications of this concept are:

   – A domain can span multiple physical locations. Since it is a logical grouping, it bears no relation to the underlying physical network.

   – A domain is a completely independent unit. If all other domains that have relationships to it were to disappear, all objects in the domain would be unaltered in the functionality.

   – A domain has its own security. This means every element of a domain must comply with its security and cannot be under the influence of the security of other domains. Each element may have security relationships to other domains.

2. Trees: One or more domains that share a contiguous namespace are called a tree. The domains in the tree are linked, from a security standpoint, by implicit Kerberos trusts. These Kerberos trusts are transitive and hierarchical. A tree is usually called by the name of the domain at its root.

3. Forests: When you have two or more trees with non-contiguous namespaces, you have a forest. Forests do not have actual names because they are just a set of cross references and Kerberos trusts, but it is usual to call them by the name of the tree that is at the root of the forest (from a Kerberos security hierarchy standpoint).

Both trees and forests share a common schema, configuration, and global catalog.

1.8.3 Physical elements: sites and domain controllers

The main physical elements of Active Directory are sites and domain controllers.

A site represents a physical location, usually with good network connectivity. A site is composed of one or more Internet Protocol (IP) subnets. The main purpose of the site is to separate the logical, administrative aspects of the network from the physical, performance-related ones, so that network administrators do not have to worry about the physical network when designing the logical domain.

A domain controller is a computer that holds a complete replica of the Active Directory subtree that represents its domain. The number and location of the domain controllers in a forest and at a given site will greatly influence the performance and reliability of the network, as well as those of any directory-enabled applications that may be in use.

It is a very simple mechanism, because (1) the DNS server will always provide the client machine with a list of IP addresses to all domain controllers, and, (2) since the client already knows its own address, it will always be able to find domain controllers in its own site (provided that they exist, of course). The client will always try those servers first. Replication traffic between domain controllers will also have different behaviors regarding scheduling and network bandwidth used, depending on how the domain controllers map to the sites.

1.8.4 Architecture

The architecture of Active Directory includes

► A data model
► Naming conventions
► Administration

Data model

The data model of Active Directory is derived from the X.500 model. The three main concepts are the schema, classes and objects.

► The schema represents the universe of all possible classes, attributes, and objects. In Active Directory implementation, the schema itself is stored as a set of classes in the database (as opposed to a flat text file, loaded at initialization time). This carries several advantages: easier access from applications, dynamic changes to the schema without the need to stop and restart the directory service, and uniform security (by using the same ACLs used for objects).

- For each class there may be mandatory attributes, additional attributes, and a list of possible parents (classes). Each class must have an OID that will identify it unambiguously. Each individual or organization that wishes to extend the schema should obtain a root OID from an issuing authority. The OIDs are represented in dotted decimal form, so, for example, every LDAP standard class starts with 2.5.6. This designation means that it is jointly defined by ISO and the ITU (2) as part of the Directory Standard (5) and that it is a class (6). The individual or organization can then manage further branches of its root OID to suit his/her/its needs.

- The objects are actual instances of the classes. As an analogy, you may think of the classes as tables in a database and the objects as the data in these tables.

Naming conventions

Some of the naming conventions used in Active Directory implementation are

- The Distinguished Name (DN) represents the full object name, from the root, such as "`CN=ADuser1 S_ADuser1, CN=Users, DC=itso, DC=ibm, DC=com`" and uses the format defined on RFC 2247. The DN is guaranteed to be unique across the forest.

- The Relative Distinguished Name (RDN) represents the portion of the DN that is an attribute of the object itself. In the preceding example, the RDN of the user is "`ADuser1 S_ADuser1`" and the RDN of the "Users" container is "CN=Users".

- The User Principal Name (UPN) is an easy-to-remember alias for the user, in the form <username>@<DNS domain name>, and very similar to the user's Internet e-mail address. It is guaranteed to be unique, but only security principals (that means objects that can be used on ACLs such as users and security groups) have UPNs. (See "Administration" on page 21 for more information about ACLs.)

- The Globally Unique Identifier (GUID) is an identifier that is statistically guaranteed to be unique for all objects in the database. All objects are identified by their GUIDs for as long as they exist. The GUID is created when the object is created. It never changes, and, when the object is destroyed, its GUID is lost forever.

- The Security Identifier (SID) is a unique identifier assigned to security principals (users and security groups). The object's SID is actually composed of two parts: the domain SID, common to all objects in a domain, and the Relative Identifier (RID), that is unique within a domain. The SID is used on all ACLs for authorization (permissions) purposes. The SID's behavior is very similar to the GUID's.

Administration

All authorization and authentication must come from a higher authority, usually Kerberos. Each object in Active Directory, including the schema objects, has a set of actions that can be performed on it called an ACL. The ACL contains sets of GUIDs and the rights associated with these, so it basically says who can do what on the object.

The ACLs are inherited down a subtree. Therefore, unless the inheritance is explicitly blocked, the ACLs will propagate down from containers to subcontainers and leaf objects. This provides for a somewhat easy form of delegation that functions by assigning ACLs to OUs.

Finally, the Directory Services Agent (DSA) manages physical storage of the objects and classes in the directory database, providing for client isolation.

1.8.5 The role of DNS

Microsoft Active Directory is heavily dependent on the DNS service for all its operations. The main roles that DNS performs are:

► **Name resolution**: DNS maps host names to IP addresses without the need for Windows Internet Naming Service (WINS): (Please note that there will still be a need for WINS for older Windows clients to access Windows 2000.)

► **Namespace definition**: The Active Directory namespace maps to the DNS namespace, simplifying the namespace design and use.

► **Locating physical components of the DNS service by using Service Record (SRV) records**: Windows 2000 clients and servers may find site and domain controller information quite easily.

1.8.6 Special roles

Although most operations on Active Directory are multi-master operations (meaning that you can read or write to any domain controller indiscriminately), a few operations need a "focal point" for greater consistency or performance reasons. These have been divided according to their scope:

► Forest-wide roles (at least one per forest)

  – Global catalog: The global catalog stores a subset of the attributes of all objects in the forest. Those are usually the attributes most used on searches. The careful placement of global catalog servers greatly improves performance, especially if you are using directory-enabled applications. You should have at least one global catalog server on each physical site.

- Schema master: The schema master is the focal point for making changes to the schema. Because this is a very infrequent and sensitive operation, Active Directory allows making changes to the schema only through the schema master. There can be only one schema master per forest.

- Domain naming master: The domain naming master is responsible for controlling the addition or removal of domains in the forest. There can be only one domain naming master per forest.

► Domain-wide roles (at least one per domain)

- RID master: RID (after domain SID)

- RID master: Responsible for allocating RIDs from the domain's RID pool. (RID is the part of the object's SID located after the domain SID.)

- Primary Domain Controller (PDC) Emulator: Services non-Windows 2000 clients for logons and password changes. The emulator also provides preferential replication of passwords on Windows 2000.

- Infrastructure Master: Updates group-to-user references when membership changes (for example, user display name).

2

Scenario 1: Domino and Active Directory

This chapter describes how to synchronize an Domino Directory and an Active Directory. We will describe some of the capabilities of the Domino 6.5 server and the new feature that enables you to synchronize the Domino Directory with Active Directory called *ADSync*. We describe in detail how to install and set up the ADSync tool. We give detailed instructions for creating users in Domino Directory using Active Directory Users and Computers Console. We also show how to register users into Active Directory from Domino.

23

2.1 Scenario 1

A fictitious company has a Microsoft Windows 2000 Advanced server that runs
Active Directory. The company also has an Domino 6.5 server that is situated on
another Microsoft Windows 2000 Server, which is their primary mail server. This
setup is shown in Figure 2-1.
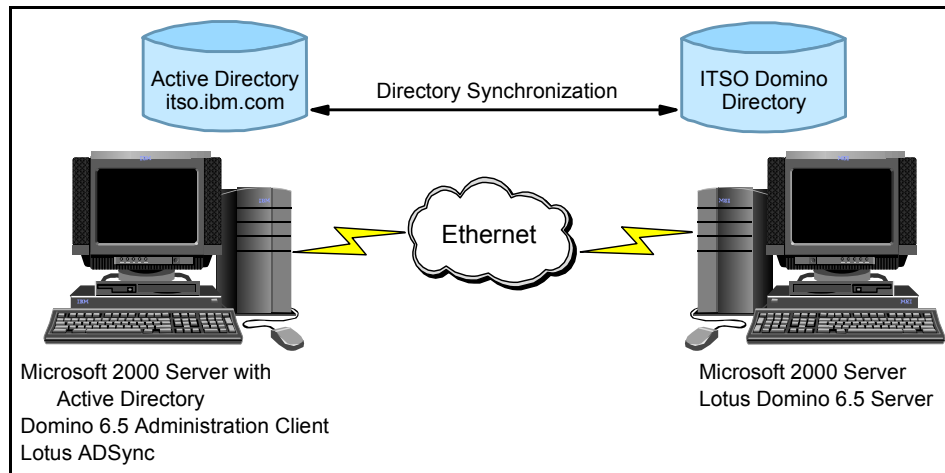
Figure 2-1   Scenario 1 setup

Note: The Domino server could be set up on any of the other platforms that
Domino supports, for example, Linux.

Every time somebody new joins the company, the administrators of each system
have to register the new person in Active Directory. They also must create a mail
file and Notes ID in Domino. Similarly, if someone leaves the company, the
administrators need to remove the entry from both directories and ensure that
the mail file is deleted. Finally, the administrators would like to be able to
synchronize the two systems to ensure that any addition, deletion, or change
made on one system is also carried out on the other system.

A new feature called ADSync was introduced in Domino 6 (and is available in
any Domino 6 or later server). This tool allows you to synchronize all (or a set of)
entries found in Active Directory with the Domino Directory. This means that you
can add, delete, or update users for both directories using a single interface.
ADSync has two interfaces that can be used, namely, the Domino Administrator
client interface and the Active Directory Users and Computers Console. (You can
use either tool depending on your comfort level with each one.) When you create
new users and groups in Active Directory, those changes are reflected in the

Domino Directory, including the creation of person or group documents, Notes IDs, passwords, and mail files for the users.

> **Note:** ADSync is a replacement for the original Microsoft Windows NT User Manager Integration function. This provided for similar operation in Domino R5 and Microsoft Windows NT 4 Server.

In order to accomplish these tasks, the Active Directory administrator must have a properly certified Notes ID and appropriate access to make changes in the Domino Directory. The registration server must be Domino 6 or later, and the Domino Administration client must be a 6 or later client. Additionally, policies must be created that contain sub-policies, either implicit or explicit, for all Domino certifiers where users will be created. Finally, you must have the appropriate rights in Active Directory to add users and groups and to synchronize passwords.

The only requirement for using the ADSync tool is to work from a workstation that administers the Active Directory and that also has the Domino 6 or later Administration client installed.

2.2 Synchronizing Active Directory and Domino Directory using ADSync

User and group accounts in Active Directory can be synchronized with the corresponding Person and Group documents in the Domino Directory. Synchronization facilitates some of the administrative functions, for example, user registration and deletion. These functions can be initiated from the Active Directory Users and Groups console in Windows or from the Domino Administration client. Users can now have common passwords for Microsoft Windows and the Domino Internet password. Synchronization copies all mapped field values from user or group objects in Active Directory, as well as member lists of the groups to corresponding documents stored in the Domino Directory.

Synchronization is initiated at these times:

► After the user or group is registered in Domino from the Active Directory Users and Groups console using ADSync

► When one or more users or groups are selected on the results pane of the Active Directory Users and Groups console and the **Synchronize with Domino** option is selected from the context menu or the toolbar

► When you change any of the properties of the user or group object and confirm your changes by clicking the **OK** or **Apply** buttons

During each synchronization, ADSync attempts to match the Active Directory object with an entry in the Domino Directory. If more than one match is found, ADSync prompts you to specify the match from those that have been located.

The Field Mappings table designates which fields to synchronize during synchronization. This setting are explained in greater detail with the other synchronization settings in "Enabling Domino Directory synchronization" on page 27.

2.3 Installing the ADSync tool

To use Lotus Active Directory Synchronization, you must install the Domino Administration client on the same workstation used to manage users and computers within your Active Directory. You also must turn on Domino Directory W2000 Sync Services during the installation of the Domino Administration client. This option is only available with the customize button during the Domino Administration client installation. The synchronization option is not selected by default; therefore, check the appropriate box.

After installing the Domino Administration client, open a DOS command prompt window, and navigate to the directory where you installed the client. Enter the following command and press **Enter**.

```
c:\Program Files\Lotus\Notes> regsvr32 nadsync.dll
```

This command adds a container entry for Lotus Domino Options to the Active Directory Users and Computers management screen and returns the confirmation shown in Figure 2-2.
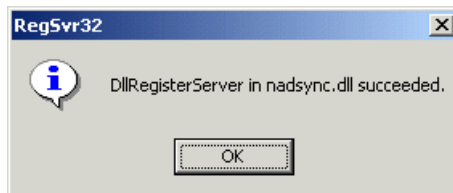
Figure 2-2   Successful registration of the ADSync DLL

You are now ready to administer users and groups in Active Directory.

Tip: To simplify the administration, copy the cert.id file from the Domino server to the data directory of your Domino Administration client.

2.4 Enabling Domino Directory synchronization

Before you start synchronizing users and groups in Active Directory and the Domino Directory, you must enable the Domino Directory Synchronization. Use the following steps to do this:

1. From the Active Directory Container shown in Figure 2-3, click the **Lotus Domino Options** entry.
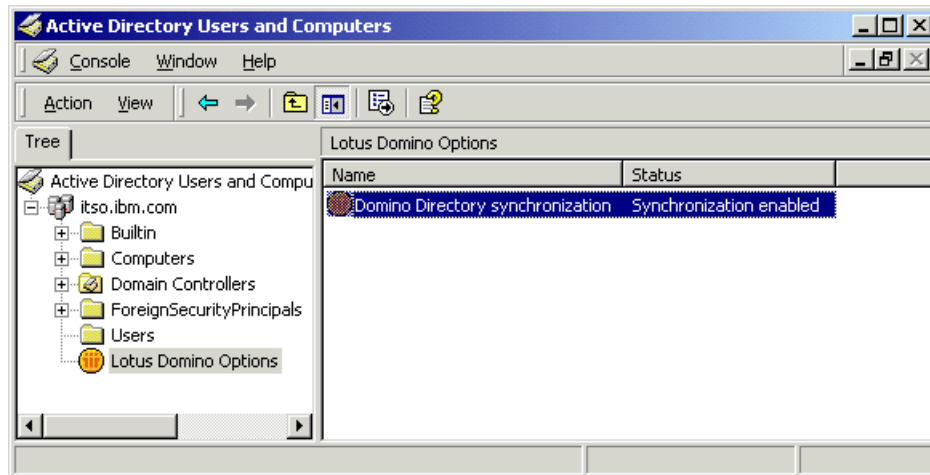
Figure 2-3   Active Directory Users and Computers

2. Double-click the entry for Domino Directory synchronization in the results pane to initialize the ADSync tool. This will require the password for the Domino administrator working from the Active Directory Users and Groups console.

3. Select an Domino server for all Active Directory/Notes user synchronizations (Figure 2-4). Then, select the appropriate Domino server from the drop-down selection box.
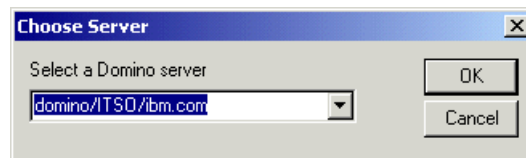
Figure 2-4   The Choose Server dialog box

4. If the initialization was successful, you should see the window shown in Figure 2-5 on page 28.
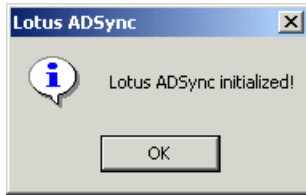
Figure 2-5   Lotus ADSync initialized successfully

With ADSync initialization complete, you have the opportunity to choose several synchronization options.

The Options window is accessible by right-clicking the **Domino Directory Synchronization** entry and choosing **Options** or simply by double-clicking the entry. The window is shown in Figure 2-6.
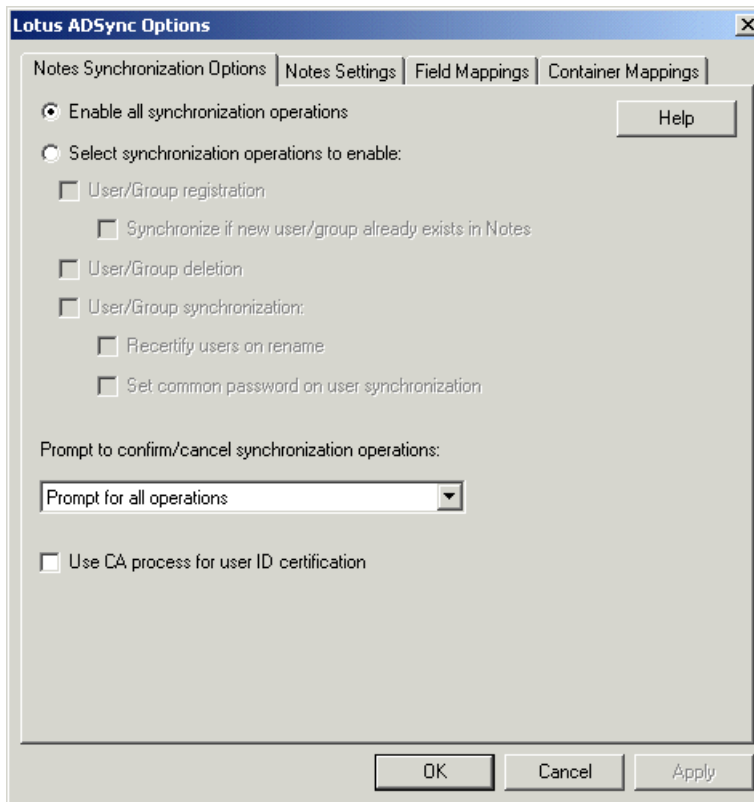
Figure 2-6   Lotus ADSync: Notes Synchronization Options tab

From the Notes Synchronization Options tab, you can

► Enable or disable all synchronization operations

► Customize synchronization options with **Select synchronization operations to enable**

► Configure prompting options from the drop-down selection box

► Choose to use the CA process for user registration

Figure 2-7 shows the Notes Settings tab.
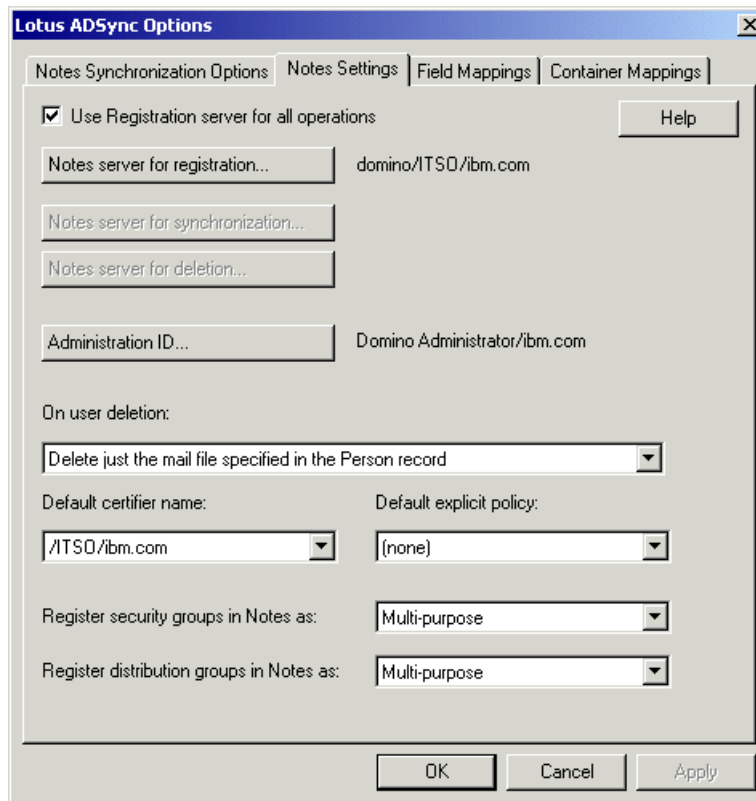
Figure 2-7   Lotus ADSync: Notes Settings tab

On the Notes Settings tab, you can specify:

► The registration server (which Domino server will be used for registration)

► The administration ID (which user ID will have administrative privileges)

► The user deletion options (accessed from the drop-down selection box and used to choose which actions should take place when a user is deleted)

► The default certifier and policy

► The group type mappings

Figure 2-8 shows the Field Mappings tab.
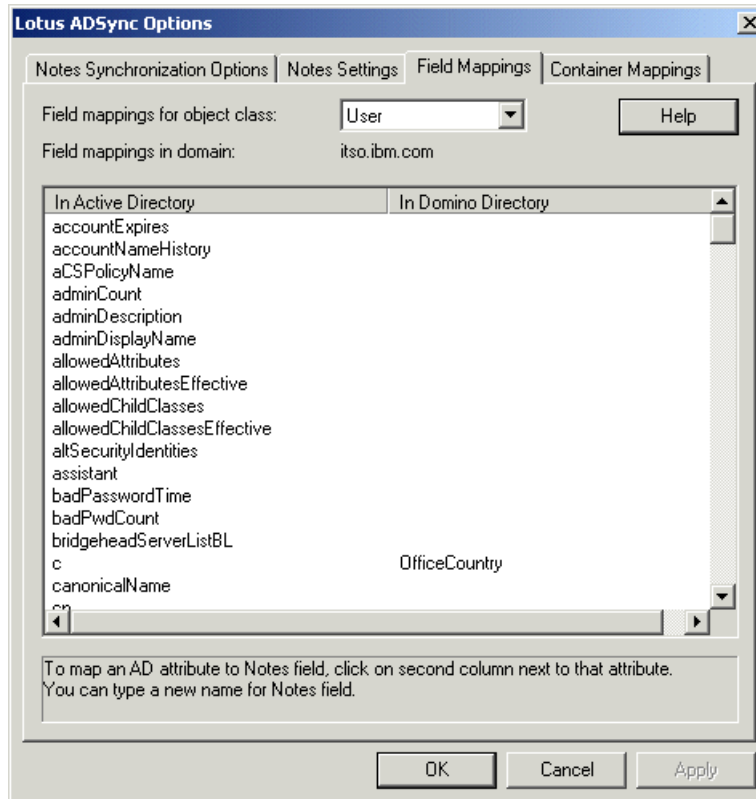
Figure 2-8   Lotus ADSync: Field Mappings tab

The Field Mappings tab, shown in Figure 2-8, is where you select which Active Directory fields are to be mapped to Domino Directory fields. During ADSync tool initialization, the schemas from Active Directory and Domino are mapped based on default settings. If additional field mappings are needed, left-click in the right column under "In Domino Directory" and a drop-down selection box with Domino directory fields is presented. This box is depicted in Figure 2-9 on page 31.
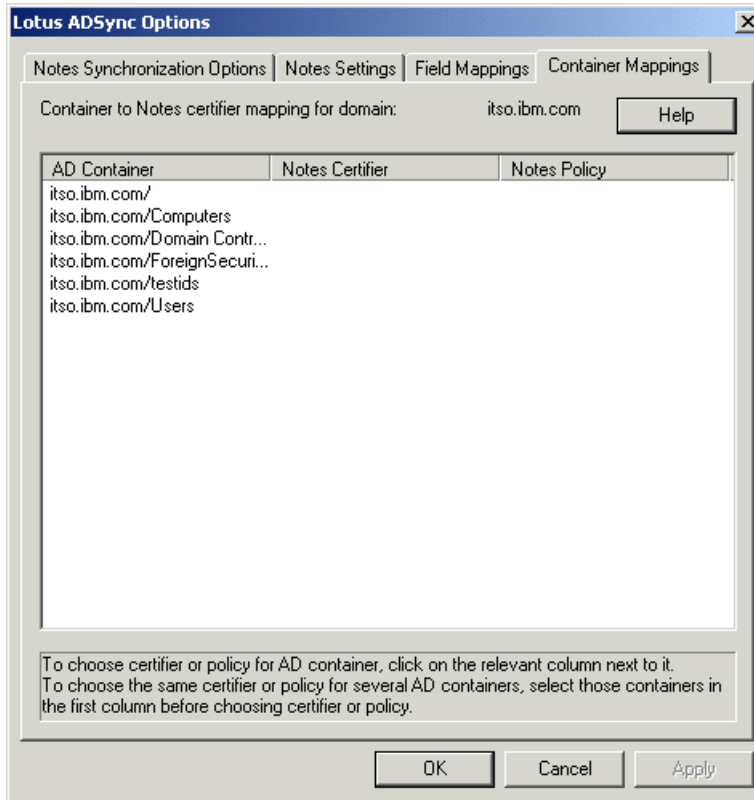
Figure 2-9   Lotus ADSync: Container Mappings tab

The Container Mappings tab, shown in Figure 2-9, is where you can map Active Directory containers to Notes certifiers and policies. Active Directory containers are a special class that has both a namespace and attributes. The container does not represent anything real or concrete but rather holds one or more objects. Objects, on the other hand, are the underlying principle of everything in Active Directory. Servers, workstations, printers, users, documents, and devices all represent objects. Each object has its own ACL and attributes.

By design, the synchronization tool allows you to preserve the hierarchies in Active Directory and Domino using mapping. You can select a specific container to map to a certifier and/or a policy. You can restrict access to a directory structure (such as a container or object) with group policies in Active Directory just as you can use the extended ACL in Domino to issue restrictions. An extended ACL is an optional directory access-control feature available for a directory created from the PUBNAMES.NTF template (that is, a Domino Directory or an Extended Directory Catalog).

The main point here is that a user can have certain rights in either directory and not the other. ADSync does not ensure that Active Directory group policies and Domino extended ACLs are synchronized. Therefore, the administrator is responsible for ensuring that no security settings are bypassed in either directory.

In the lab, we select the container root, the domain controllers, and the Users container. Beside the container you wish to associate with a certifier, double-click in the Notes Certifier column to see your selection choices. Select the appropriate certifier and click **OK** to continue.

Once we decide upon our synchronization settings, we can begin creating users and groups from the Active Directory console.

2.5 Registering users of Active Directory

To access Active Directory Users and Computers from your Windows workstation, click **Start** → **Programs** → **Administrative Tools** → **Active Directory Users and Computers**. Domino users and groups are created by one of two methods:

1. In the left pane, right-click an entry and choose your action from the pop-up menu.

2. In the results pane, select one or more users and groups, then select **Register in Domino** from either the context menu, the toolbar, or by right clicking the entry and using the pop-up menu.

2.5.1 Registering existing Active Directory users in Domino

If ADSync is set up correctly, you will be able to register existing Active Directory users as Domino users. You can also create new users in Active Directory and choose to register them in Domino at the same time.

To register existing Active Directory users in Domino, select the appropriate container in the left-hand pane, then choose which user or group to register in the right-hand pane. Right-click the selected entry. A pop-up window is presented with **Register in Domino** as one of the options. This is shown in Figure 2-10 on page 33.
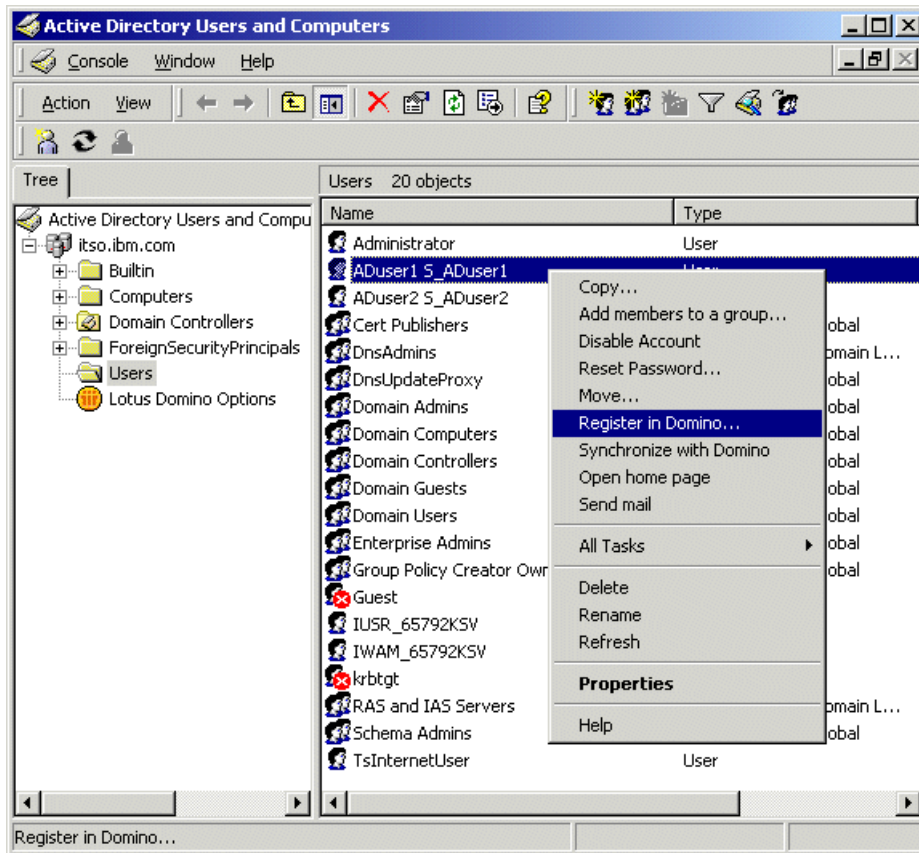
Figure 2-10   Pop up showing the Register in Domino option

Select **Register in Domino** and the next window will present a number of
registration options to you. These are shown to you in Figure 2-11 on page 34.
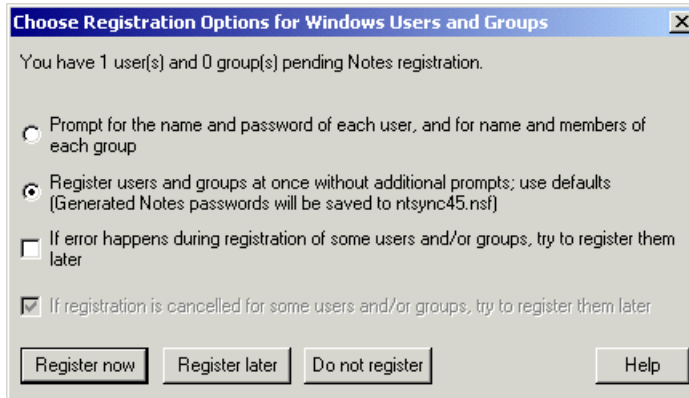
Figure 2-11 Registration options for Windows Users and Groups

The quicker and easier way to continue is to select the second option, **Register users and groups at once without additional prompts; use defaults**. After you click **Register now**, ADSync will register the Active Directory user in Domino, using default settings. If there are no errors, you receive a message informing you that the user has been successfully registered in Notes. This can be seen in Figure 2-12.
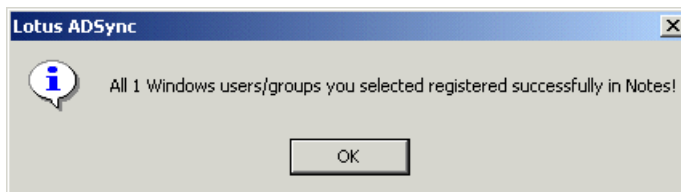
Figure 2-12 A successful registration of a user in Notes

To check whether the person was registered successfully in Domino, open up the Lotus Domino Administration client. Ensure that you have your Domino server open, and then click the **People & Groups** tab. Select **People** on the left-hand side. You should see the person you added from Active Directory in the list of Domino users as shown in Figure 2-13 on page 35.
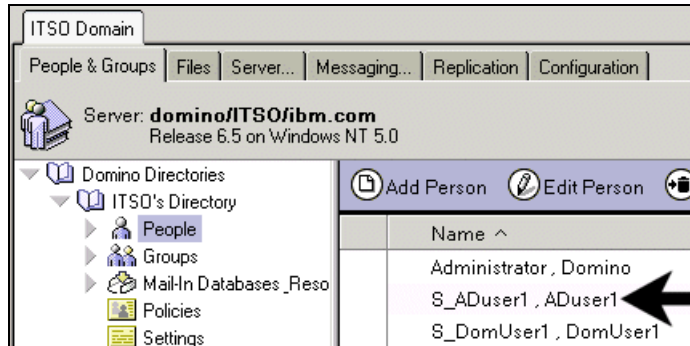
Figure 2-13   The Active Directory user registered in Domino

Alternatively, if you wish to use settings that are not the default value, select the first option, **Prompt for the name and password of each user, and for name and members of each group**, as shown in Figure 2-14.
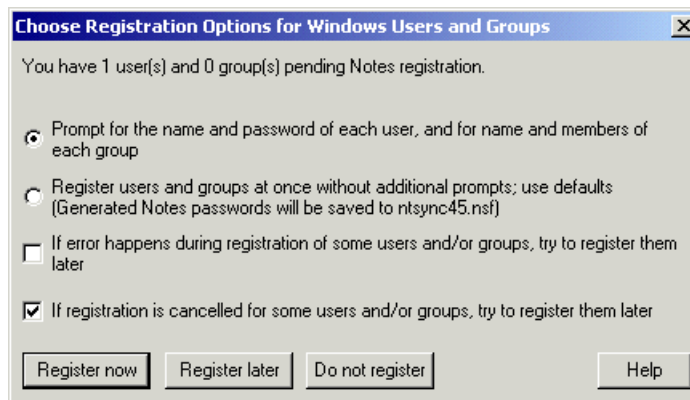
Figure 2-14   Alternative option for registering a AD user in Domino

Selecting this option will display another window where you can change the settings for each user that you register. This window is shown in Figure 2-15 on page 36.
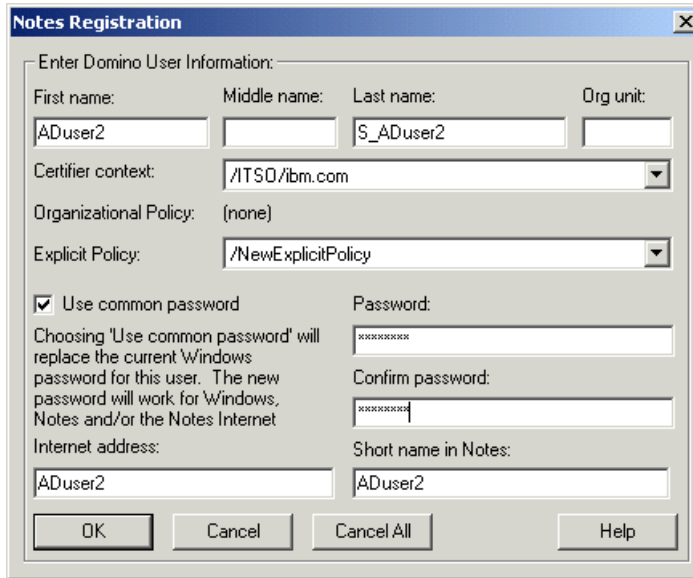
Figure 2-15   The Notes Registration dialog

Ensure that the fields are correct, and then enter the new password for the user.

Note: A common password means that the user will have the same password for his or her Microsoft Windows domain, IBM Lotus Notes e-mail, and Notes Internet password. The password entered here will overwrite the previous Microsoft Windows password.

Click **OK.**

You may be prompted for the certifier location and password, as shown in Figure 2-16. Enter the required information and click **OK**.
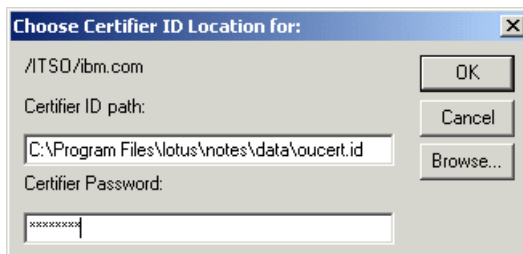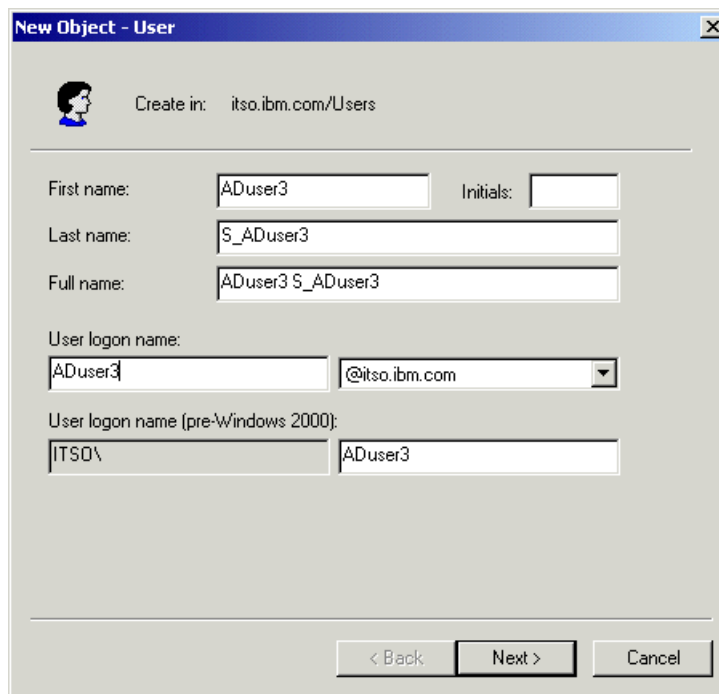
Figure 2-16   The Certifier ID location window

The Lotus ADSync will now connect to the specified Domino server and register the Active Directory user as a new Notes user.

2.5.2 Registering new users in both Active Directory and Domino

To illustrate how to create a new user or group in Active Directory and register them in Domino, we can create a new user account in Active Directory by clicking the **New User** icon in the Active Directory toolbar. You can also use the Action drop-down menu for this option.

The first window for New Object - User will be returned, as shown in Figure 2-17. After entering the data for the appropriate fields, click **Next** to continue.

Figure 2-17   User information window

Enter the information for the password fields and click **Next** to continue. Your choices for password expiration and modification, as well as disabled accounts, are based on your company's security policies. This User password version of the window is shown in Figure 2-18 on page 38.

Figure 2-18   User password window

In the window shown below, you will notice an option to register this user or group in the Domino Directory. This window also provides fields for choosing the certifier context, an explicit policy, password fields for Domino, Notes short name, and an Internet address and the ability to enable the use of common passwords. Once you have supplied the necessary information, click **Next** to continue. The Domino information version of the window is depicted in Figure 2-19 on page 39.

Figure 2-19   Domino information window

The new user creation process then presents you with a summary of the user object you are about to create, as shown in Figure 2-20 on page 40.
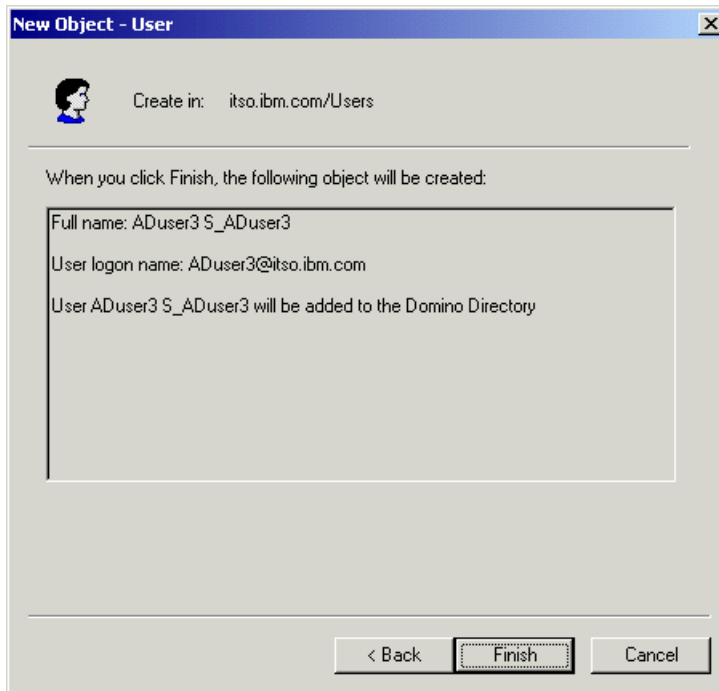
Figure 2-20   Confirmation of the settings for the new user

Click **Finish** and the system will generate the Active Directory object, the new person document in the Domino Directory, a Lotus Notes ID file, and a user mail file.

That's it! You have successfully created a new user from within Active Directory and in doing so, you generated new objects for that person in both Domino and Microsoft Windows 2000. You can again check to see whether the person has been registered as a Notes user by opening up the Lotus Administration client. The name should appear again in the Person view as shown in Figure 2-21 on page 41.
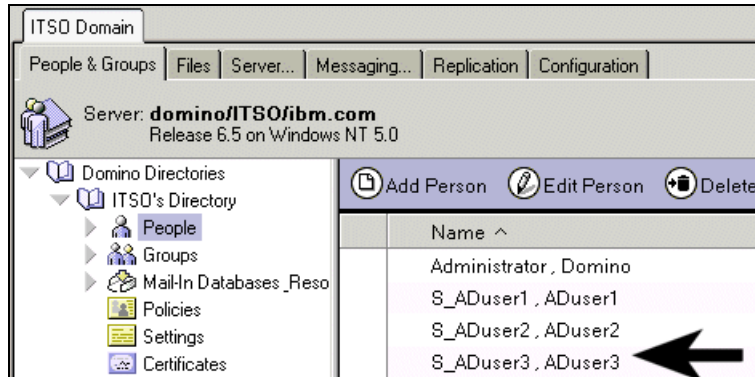
Figure 2-21   New Active Directory user added shown in Domino Administrator

2.6 Registering users using the Domino Administrator client

In addition to registering users and groups from the Active Directory Users and Groups console for both the Microsoft Windows 2000 and the Domino environments, you can register them from the Domino Administrator client.

> **Note:** ADSync provides the ability to create *new* users in both Domino and Active Directory simultaneously. It not possible to register existing Domino users as new Active Directory users.

Using the Domino Administration client, select the server to be used for registration. There are two possible ways to register a new person in the Administrator client:

1. Select the **People & Groups** tab and select **Tools** → **People** → **Register** on the right side of the screen.

2. Select the **Configuration** tab and select **Tools** → **Registration** → **Person**.

If this is the first time you are registering a new person in Domino, the administration client then prompts you for the Notes Certifier ID file. Ensure that the correct server and certifier file have been chosen, as shown in Figure 2-21. If this information is correct, click **OK**. Then, supply the certifier password and click **OK** again.

The Domino Administration client then presents you with a Register Person window. Complete the registration fields in this window, and then click the check

box for **Advanced** options. The Advanced option is circled in Figure 2-22 on page 42.
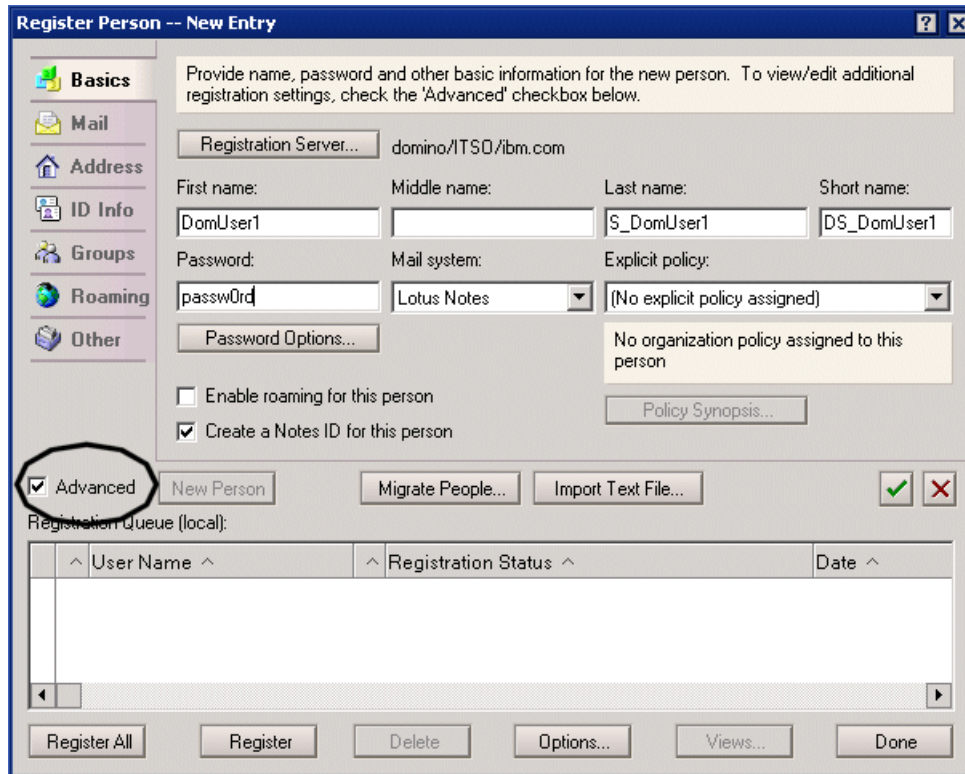
Figure 2-22   The Register Person window with Advanced options circled

Complete the information appropriate for your organization in the Mail, Address, ID Info, Groups, and Roaming sections. Click the tab for the **Other** section; click the **Windows User Options** button to add this person to Windows 2000, as shown in Figure 2-23.
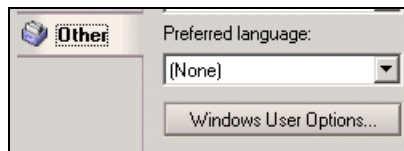
Figure 2-23   The Windows User Options button on the Other tab

In this window, select the Active Directory container and Microsoft Windows 2000 groups to add this person to. Then click **OK** when finished. This particular

account is placed in the Users container. We can place the user in any container appropriate for that account's security rights. See Figure 2-24 on page 43.
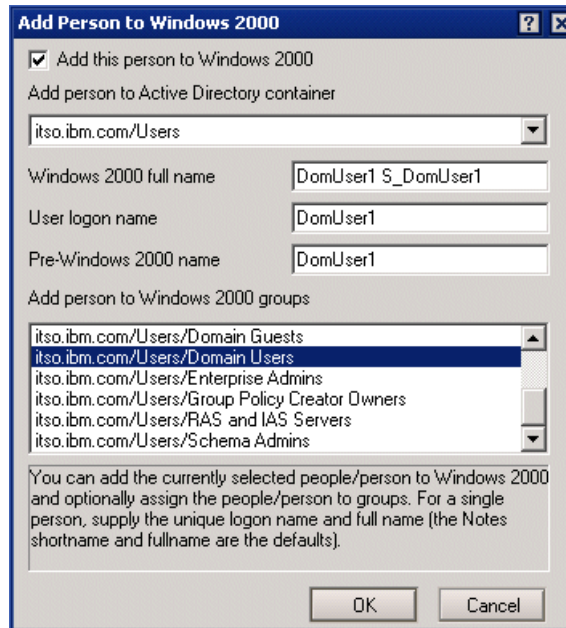
Figure 2-24   The Add Person to Windows 2000 window

Finally, click the check mark box in the Register Person window to confirm you have finished entering all necessary data for this person. (The check mark box is located on the right-hand side above the Registration Queue block.) See Figure 2-25.
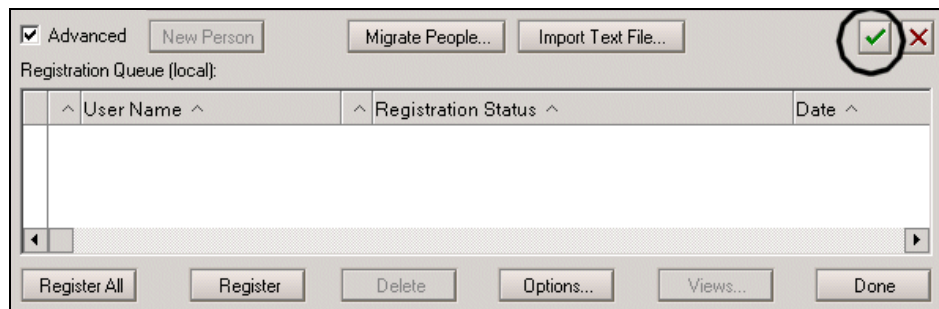
Figure 2-25   The check mark confirms the person's registration

The entry will then be added to the Registration Queue window at the bottom of the screen. Click **Register** to initiate the registration process, as shown in Figure 2-26 on page 44.
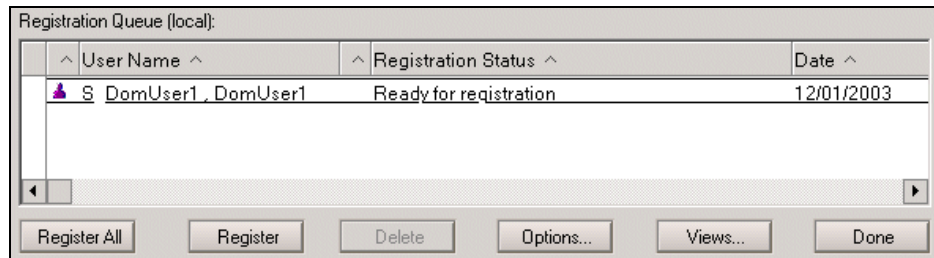
Figure 2-26   The registration queue

Once the registration process completes, this person will exist in both the Domino Directory and Active Directory. Then, you will be informed that the person was registered successfully, as shown in Figure 2-27.
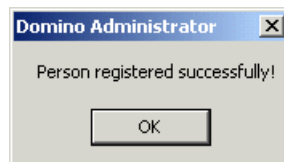
Figure 2-27   Person registered successfully in Lotus Administration client

We can now check whether the person has been added correctly to Active Directory. To access Active Directory Users and Computers, click **Start** → **Programs** → **Administrative Tools** → **Active Directory Users and Computers**. Select the **Users** (or the container you specified in the Lotus Administration client). You can see the name of the user you have just created in the Lotus Administration client, as is shown in Figure 2-28 on page 45.
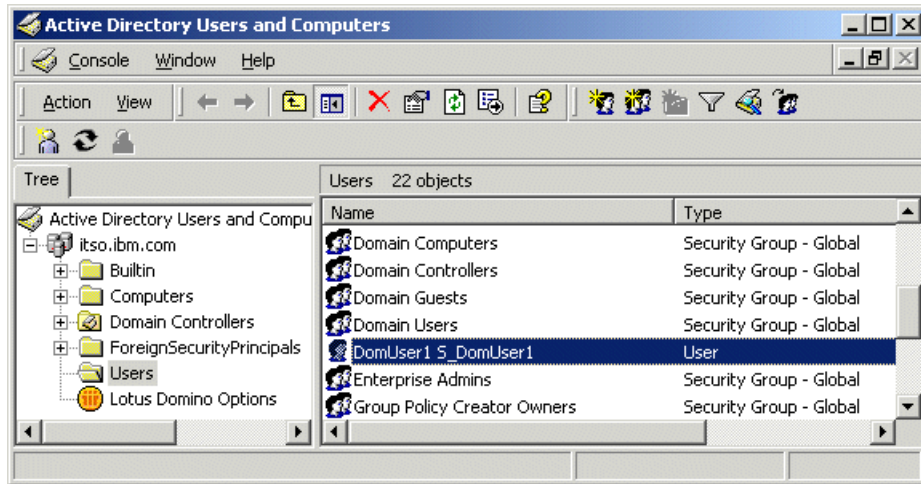
Figure 2-28   The user created in the Lotus Administration client reflected as an AD user

2.7 Deleting users

When you delete a user or group from Active Directory and there is a corresponding user or group in the Domino Directory synchronized with it, ADSync removes the Person document or Group document for that Domino Directory entry using the Administration Process on the deletion server. You can designate a deletion server and change user mail file deletion settings in the Notes Settings tab of the Lotus ADSync Options dialog box.

When you delete a Notes user or group, all references to it are removed from the Domino Directory by the Domino Administration Process running on a Domino server. After initiating the deletion, you must approve the request in the Administration Requests (ADMIN4.NSF) database on the Domino server.

2.7.1 Deleting users with Active Directory

To delete users from both Active Directory and Domino using Active Directory, click **Start** → **Programs** → **Administrative Tools** → **Active Directory Users and Computers**. In the Microsoft Management Console, select the required container in the left hand panel. On the right-hand panel, find the user you wish to delete. Right-click the name of this user and select **Delete** from the pop-up window, as shown in Figure 2-29 on page 46.
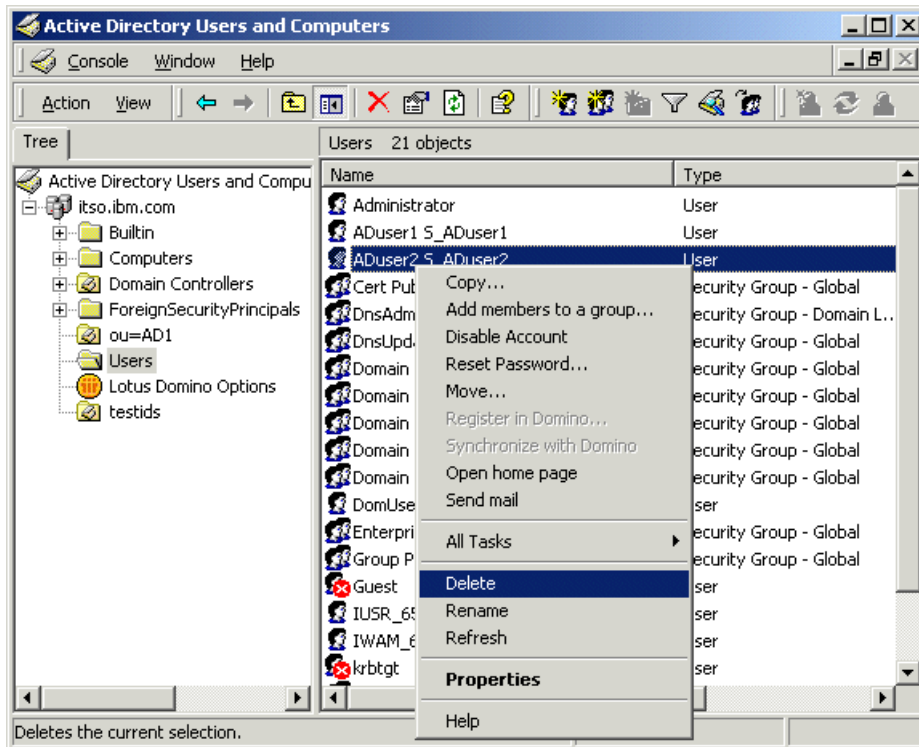
Figure 2-29   Deleting users using Active Directory

Click **Yes** to confirm the deletion of this user.

2.7.2 Deleting users with the Domino Administrator client

From the Domino Administration client, select the **Peoples & Groups** tab. Select the People view. In the right-hand view, select the person you wish to delete (you may choose more than one person). Click the **Delete Person** action button at the top of the view. A **Delete Person** window will now appear, as shown in Figure 2-30 on page 47.
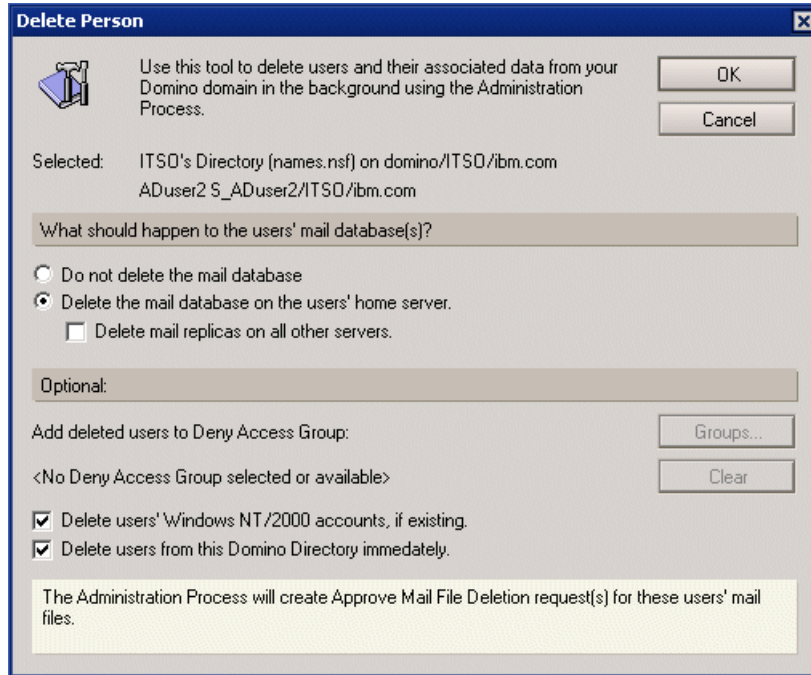
Figure 2-30   Deleting users using the Domino Administrator client

Place a check mark in the box adjacent to **Delete users Windows NT/2000 accounts, if existing**. Click OK and the person will be deleted both from Domino and Active Directory.

2.8 Data synchronization using Directory Integrator

It is also possible to use Directory Integrator (ITDI) to create new accounts and synchronize data between Domino and Active Directory. The major advantage that Directory Integrator has over ADSync is that it can be set up in either a triggered or batch execution mode. This differs significantly from ADSync, as ADSync requires manual intervention. This means that you can automate the management of user accounts in either system (as well as others) using Directory Integrator.

More information about integrating Domino users with Directory Integrator can be found in the *IBM Tivoli Directory Integrator 5.2: Reference Guide*, SC32-1377 located at:

> http://publib.boulder.ibm.com/tividd/td/IBMDirectoryIntegrator5.2.html

information about using Active Directory with Directory Integrator can also be found in the *IBM Tivoli Directory Integrator 5.2: Reference Guide*, SC32-1377 located at:

`http://publib.boulder.ibm.com/tividd/td/IBMDirectoryIntegrator5.2.html`

There are also Password Synchronization plug-ins available for Directory Integrator. These plug-ins intercept any password changes to user accounts on various directories, including Domino, Active Directory, and Directory Servers, and update the corresponding fields in the required places.

More information about Password Synchronization Plug-ins for Directory Integrator can also be found in the *IBM Tivoli Directory Integrator 5.2: Reference Guide*, SC32-1377 located at:

`http://publib.boulder.ibm.com/tividd/td/IBMDirectoryIntegrator5.2.html`

We discuss the setup and use of Directory Integrator with examples and in much greater detail in Chapter 3, "Scenario 2: Directory Server and Active Directory" on page 49. Other scenarios using Directory Integrator are covered in Chapter 4, "Scenario 3: Directory Server and Domino" on page 75 and Chapter 5, "Scenario 4: Integrating data from three different sources" on page 105.

3

Scenario 2: Directory Server and Active Directory

This chapter describes the following:

► Installation and Configuration of Directory Integrator

► Migration of user data from Directory Server to Active Directory using Directory Integrator

► Migration of user data from Active Directory to Directory Server using Directory Integrator

► Synchronization of changes in Active Directory with Directory Server using Directory Integrator

► Two-way synchronization between Directory Server and Active Directory using Directory Integrator

3.1 Introduction

The chapter describes how to synchronize user data between Directory Server and Active Directory using Directory Integrator. The chapter starts with a brief description of the scenario. The chapter then describes the configuration of Directory Integrator and process of using Directory Integrator to migrate the users from Directory Server to Active Directory and vice versa. The synchronization of entry changes in Active Directory with Directory Server is described later. The chapter concludes with two-way synchronization between the two Directory Servers.

3.2 Scenario 2

The following scenario focuses on two organizational units in a fictional company. One organizational unit uses Directory Server as its directory, while the second organizational unit uses Active Directory as its directory. There are no common records in the two directories. The organization decides to integrate the data and use Active Directory as the primary directory for the entire organization.

The user data from the Directory Server will be migrated to the Active Directory; then the data from Active Directory will be migrated to the Directory Server. A policy decision has been taken to use Active Directory as the directory for the entire organization. Any user account addition, modification, and deletion thereafter will be made to the Active Directory only. However, it is decided to keep the Directory Server functional only for searches. Therefore, the data in the Active Directory needs to be synchronized with Directory Server 5.2.

Directory Integrator 5.2 is be used to do the synchronization.

3.2.1 The Test Environment

The test setup consists of an isolated network, with domain itso.ibm.com, having two Microsoft Windows 2000 Server machines. One of the machines, host name m23wpk40 and IP address 10.0.0.1, has Active Directory running on it. This machine is the domain controller and also has Domain Name Service (DNS) configured and running.

The other machine, host name m23wpk60 and IP address 10.0.0.3, has Directory Server 5.2 running on it. Figure 3-1 on page 51 shows the Scenario Two setup.
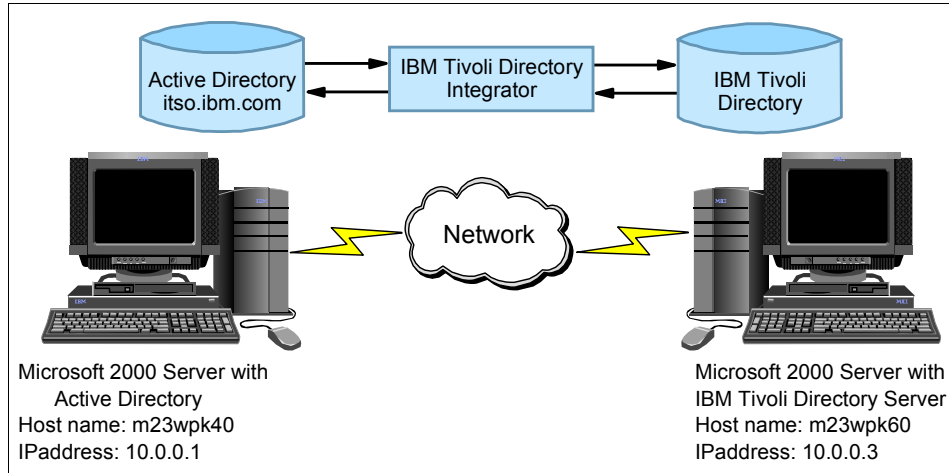
Figure 3-1 Scenario Two setup

3.2.2 The testing procedure

To keep the demonstration simple, the following start-up configuration settings are used:

▶ Both the Active Directory and the Directory Server have only user records. The RDN for the user data is as follows:

For Directory Server: `ou=IDS1,ou=itso,o=ibm,c=us`

For Active Directory: `ou=AD1,dc=itso,dc=ibm,dc=com`

▶ The changelog for the Directory Server has been enabled.

▶ Directory Server changelog contains records with changetype *add* only. Similarly, all the changes made to the Active Directory are user additions only.

The scenario will be built step-by-step. We start with the migration of data from Directory Server to Active Directory and vice versa, using Directory Integrator. Then the cases of user addition, modification, and deletion are covered.

▶ The data from Directory Server will be migrated to Active Directory under `ou=IDS1,dc=itso,dc=ibm,dc=com`

▶ The data from Active Directory will be migrated to Directory Server under `ou=AD1,ou=itso,o=ibm,c=us`

After the user migration is complete, the Active Directory is used as the organizational directory server.

► Any user addition, modification, and deletion is synchronized with the Directory Server.

3.3 Installation and configuration of Directory Integrator

For an introduction to the Directory Integrator, see 1.5, "IBM Tivoli Directory Integrator" on page 12.

3.3.1 Installation of Directory Integrator

1. On the installation CD (or wherever the Directory Integrator package is downloaded), locate setupwin32.exe. Double-click it, or type `setupwin32.exe` at the command prompt.

2. The program loads the InstallShield and starts the installation. The first window gives you information about the product. Click **Next** to continue.

3. The next window is the license agreement. On accepting the license agreement, the **Next** button will be enabled. Click **Next** to continue.

4. The next window displays a default working directory where the product is installed. You may change this location by clicking **Browse**. Click **Next** to continue. See Figure 3-2.
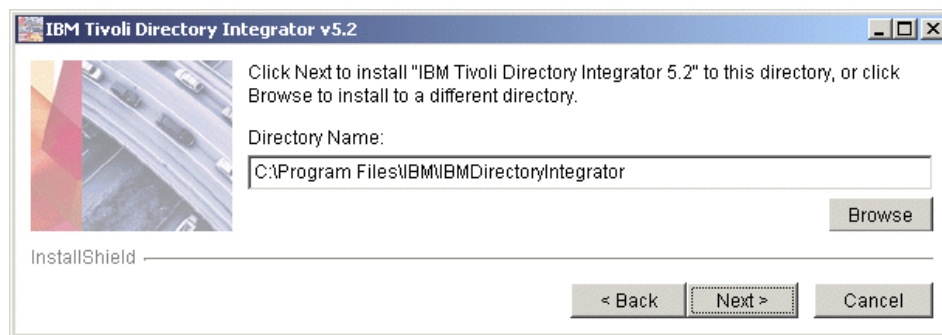
Figure 3-2 Directory Integrator Installation

5. The next window provides the information about where the product is installed and the disk space that the product requires.Click **Next** to continue.

6. The product installation begins. On completion, the final window is displayed. Click **Finish**.

3.3.2 Configuration of Directory Integrator

The procedure briefly explains the customized configuration of Directory Integrator for the case at hand, which is the migration of data from Directory Server to Active Directory. For details of the configuration, refer to the *Directory Integrator Getting Started Guide*, SC32-1382, which can be found at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryIntegrator5.2.html

Configuration of an AssemblyLine to Migrate data from Directory Server to Active Directory

1. Locate the ibmditk.bat file in the IBM Directory Integrator installation directory. Invoke Directory Integrator by running the ibmditk.bat file.

2. To create a new configuration file, click **File** → **New**. Input a valid file name of the configuration file, as shown in Figure 3-3. Click **OK**.
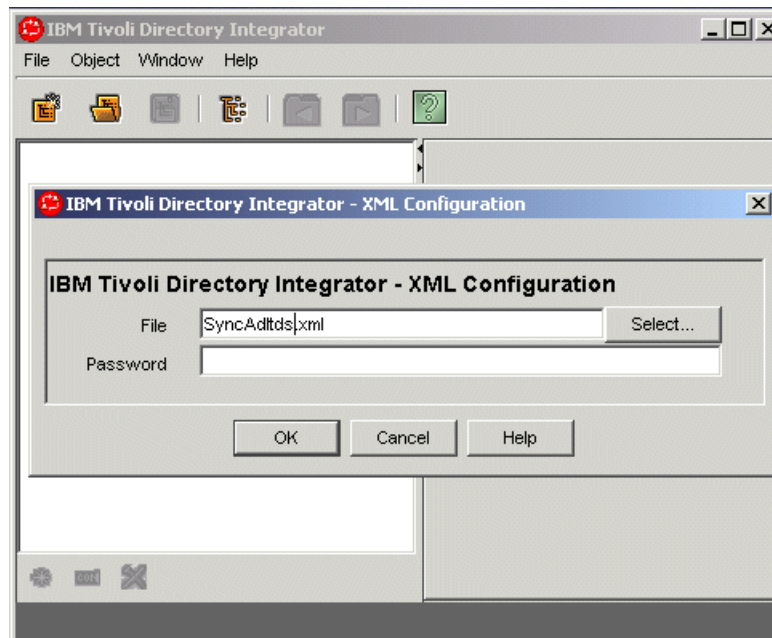
Figure 3-3   Creating new configuration file

3. To create a new AssemblyLine, right-click **AssemblyLines** in the left-hand pane. Click the **New AssemblyLine** icon in the window that appears. Input a valid name in the text area, as shown in Figure 3-4 on page 54. Click **OK**.
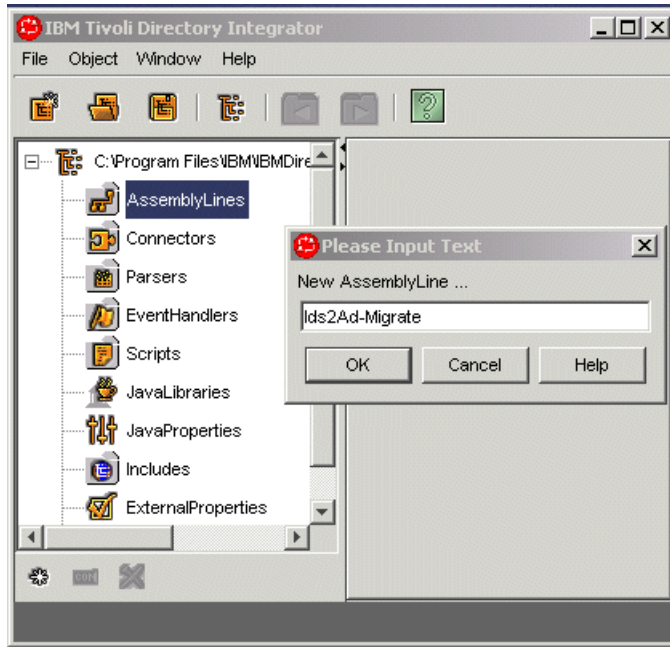
Figure 3-4   Create new AssemblyLine

4. Click the newly-created **AssemblyLine**. The right pane will now display the Data Flow tab. Locate the **Add Connector** button at the bottom of the window (circled in Figure 3-5 on page 55). Move the cursor to the button and hold. The tool tip provides useful information. Click the **Add Connector** button.
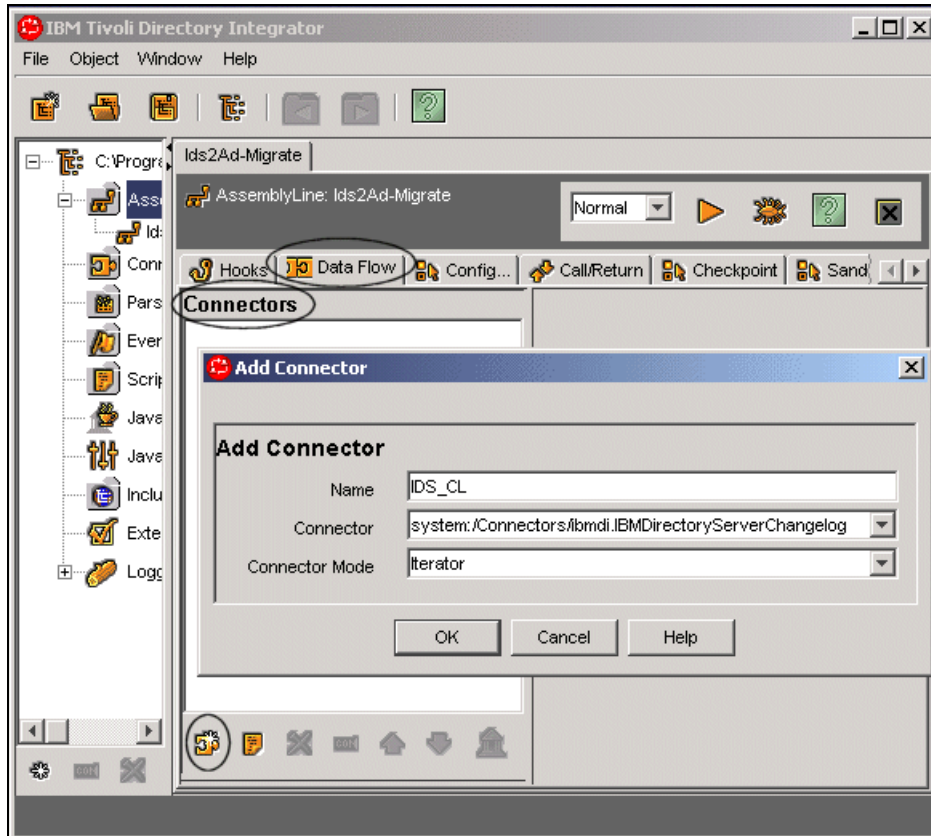
Figure 3-5   Add connector

5. The Add Connector window appears. Input a valid name for the connector. In the Connector drop-down list, locate the IBMDirectoryServerChangelog. The Connector mode is Iterator by default. Click **OK**. See Figure 3-5.

6. Set up the configuration of the connector. For the details of connector configuration, refer to *Directory Integrator's ReferenceGuide*, SC32-1377. Look under: **Connectors** → **LDAP Connector** → **IBM Directory Changelog Connector**. The reference guide can be found at:

   http://publib.boulder.ibm.com/tividd/td/IBMDirectoryIntegrator5.2.html

   See Figure 3-6 on page 56.
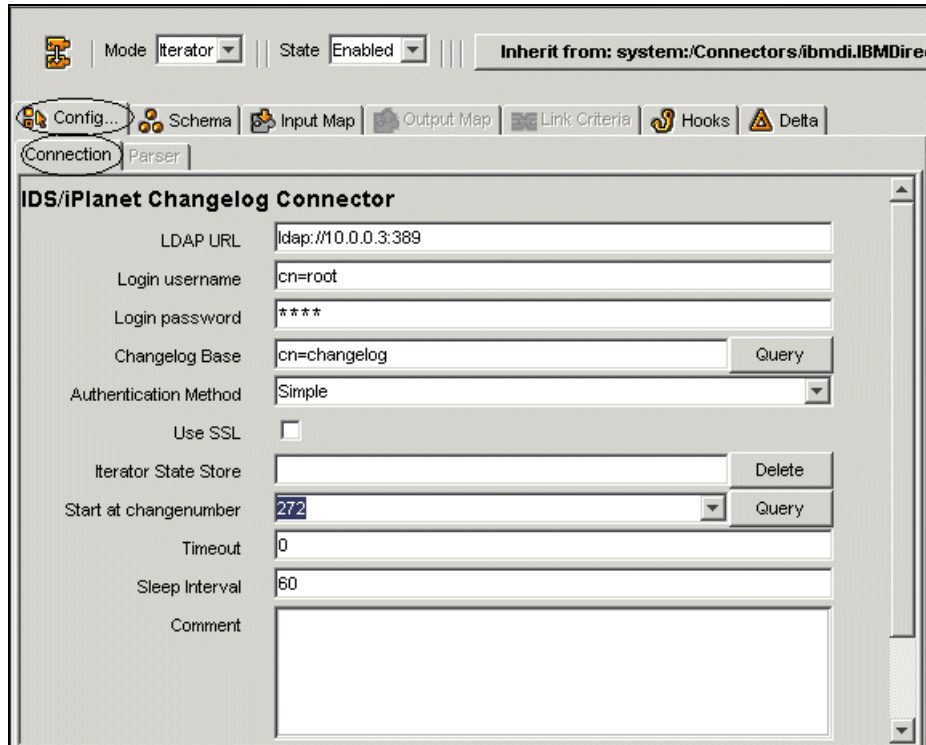
Figure 3-6   Configuration of Directory Server Changelog connector

This completes the configuration of the added connector. The attribute mapping procedure is now explained.

To view the attributes present in Directory Server, do the following:

1. Click the **Schema** tab.

2. Click the **Connect to the data source** icon (circled in Figure 3-7). If the configuration is correct and the datasource (Directory Server) is up and running, the connection with the datasource is established. See Figure 3-7.
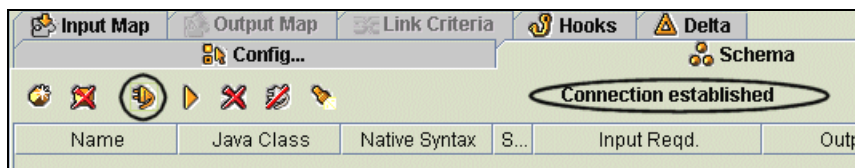
Figure 3-7   Connect to the datasource

3. Click the **Read the next entry** button (the arrow icon circled in Figure 3-8). An entry in the datasource will be displayed. Figure 3-8 shows a few sample attributes retrieved. In order to retrieve all the available attributes, you probably need to browse a few more entries in the Schema Tab until you can see the expected attribute showing there. (If you know the expected attribute's name, you can always add it manually.)
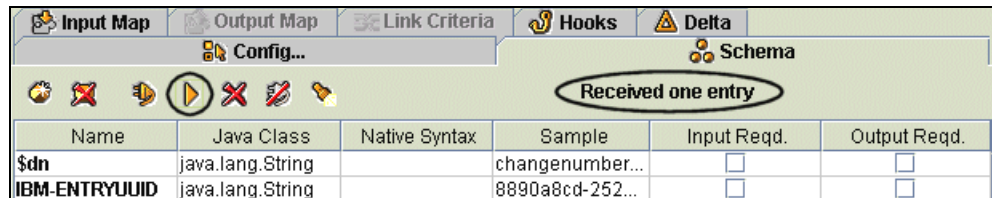
Figure 3-8   View entries

With the attributes retrieved, the input map can now be formed:

1. Click the **Input Map** tab. The attributes available in the connector are displayed in the bottom right pane.

    Now depending upon the case, either a few or all the attributes could be mapped into the Work Entry that is formed. The attribute could be mapped either by (1) dragging and dropping the attributes from the available connector attributes to the work entry or (2) adding the attributes explicitly and mapping those as per the requirement. For simplicity, all the attributes are mapped as shown in Figure 3-9 on page 58.
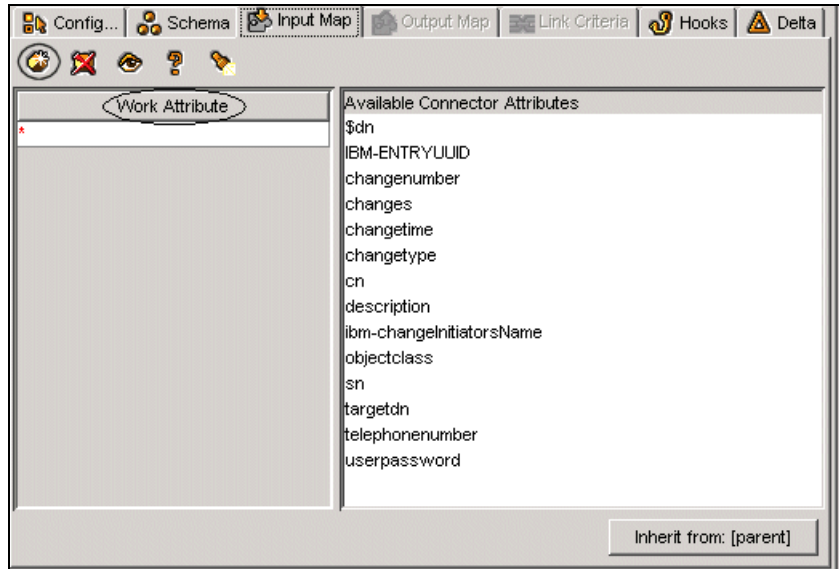
Figure 3-9   Input map creation

2. Click the **Add a new attribute to the Attribute map** button on the top of the Work Attribute window.

3. In the text area type an asterix ("*****") and click **OK**.

   By the mapping above, all the attributes of an entry in Directory Server will be available in the work object.

Now the task is to map the attributes to Active Directory attributes and add the mapped entry into Active Directory. The task could be achieved in many ways. Here the Default Success hook is used to add the attribute distinguishedName to the work entry by writing a script.

1. Click the **Hooks** tab of the connector. A list of hooks will be shown in the middle pane. Locate the Default Success hook.

2. Add the script as shown in Figure 3-10 on page 59 to build the distinguishedName attribute.
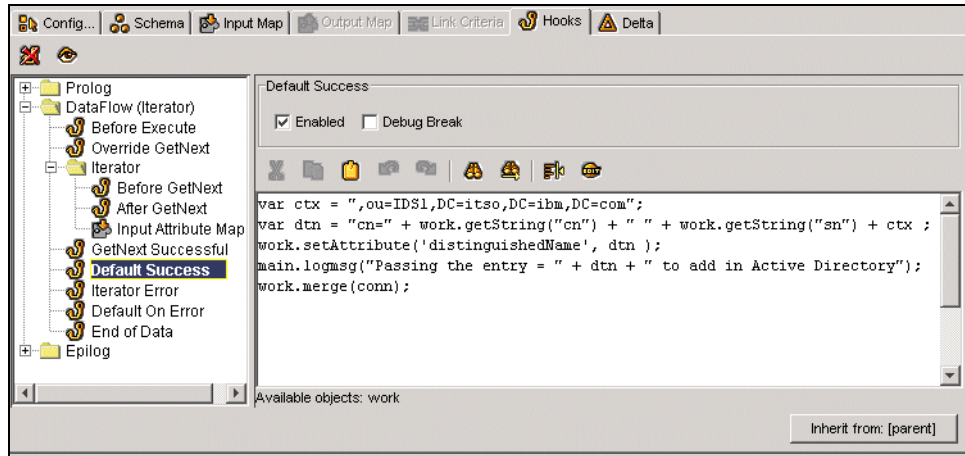
Figure 3-10   Hook with attribute mapping code

The translation of distinguished name is done as follows:

cn=<*cn of some user*>,ou=IDS1,ou=itso,o=ibm,c=us

of the entry in Directory Server to

cn=<*cn of some user*> <*sn of some user*>,ou=IDS1,dc=itso,dc=ibm,dc=com

to the entry in Active Directory.

> **Note:** The basic configuration steps for a connector have now been explained. For the remainder of this chapter, the steps carried out are mentioned directly. Any new configuration aspect or detail is only described and explained.

1. An LDAP Connector to add the entries into Active Directory is added by clicking the **Add Connector** button. Figure 3-11 on page 60 shows the configuration of the connector.
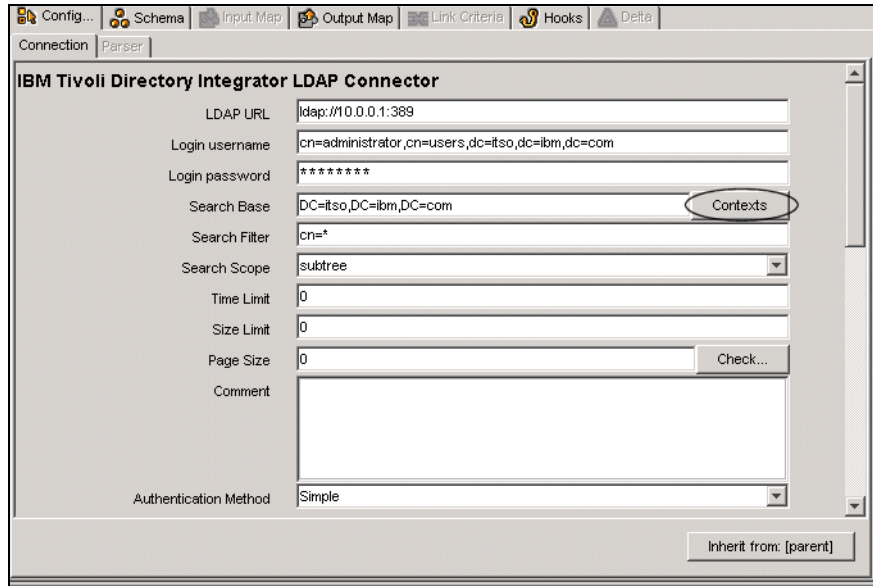
Figure 3-11   Configuration: LDAP Connector

The search base can be selected by clicking the **Contexts** button in the right-hand pane. The drop-down list contains all of the valid Contexts.

The mapping of the attributes is crucial. It depends upon the schema and the type of data contained in the directory. In this case, we map the Active Directory attributes to Directory Server attributes using the following process:

1. Set the attributes as follows:

   – **$dn**: distinguishedName (formed in the Default Success Hook

   – **description**: description

   – **displayName**: combination of `cn` and `sn`: `cn` + " " + `sn`

   – **distinguishedName**: distinguishedName

   – **givenName**: `cn`

   – **objectclass**: multivalued with values: top, person, organizationalperson and user

   – **sAMAccountName**: combination of `cn` and `sn`: `cn` + "_" + `sn`

   – **sn**: sn

   – **telephoneNumber**: telephoneNumber

   – **userPrincipalName**: `cn` + "_" + `sn` + "@itso.ibm.com"

2. Click the **Output Map** to map the attributes. The attributes can be dragged and dropped from the Work entry formed or formed by clicking **Add attribute** and then mapping it as per the requirement. Figure 3-12 shows the attribute map configured with a sample mapping.
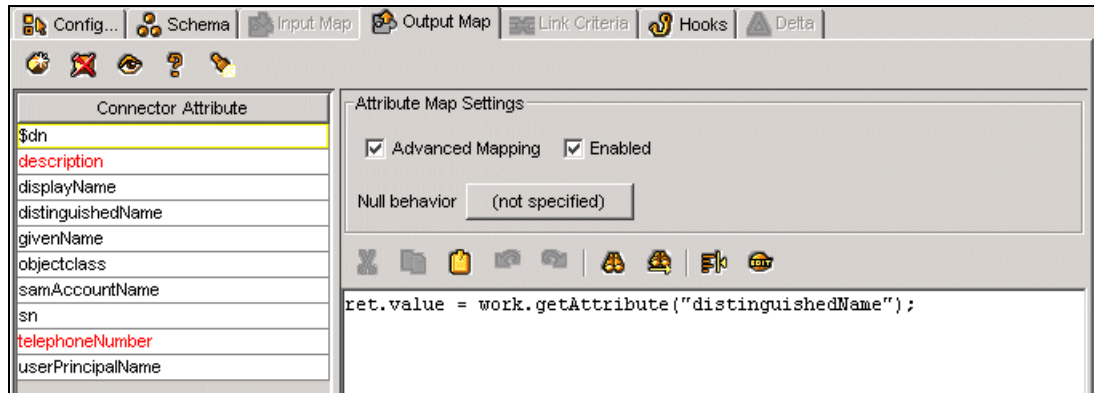
Figure 3-12   Output Map

The procedure above is be sufficient to add an user in the Active Directory. But the Unicode Password of the user in Active Directory is not set (even though the userpassword is mapped). This setting requires an SSL connection with the Active Directory. To achieve this, another LDAP Connector in Update mode working over SSL is added. For the SSL setting of Active Directory and Directory Integrator, refer to the *Directory Integrator User's Guide*, SC32-1378, which can be found at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryIntegrator5.2.html

Figure 3-13 on page 62 shows the configuration of the connector. Also, select the **Use SSL** and **Auto Map AD Password** check boxes on the configuration window.
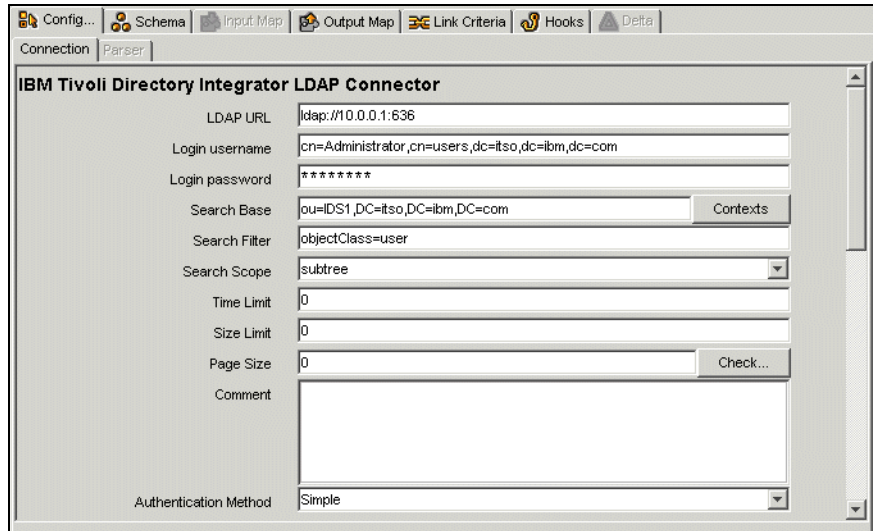
Figure 3-13   Configure LDAP Connector for SSL

To update an entry, a Link Criteria needs to be specified, as follows:

1. Click the **Link Criteria** tab of the connector. Click the **Add New Link Criteria** button.

2. The $dn is the Connector Attribute chosen to search for the entry. The value of the attribute is picked from the value of $distinguishedName attribute, as shown in Figure 3-14.
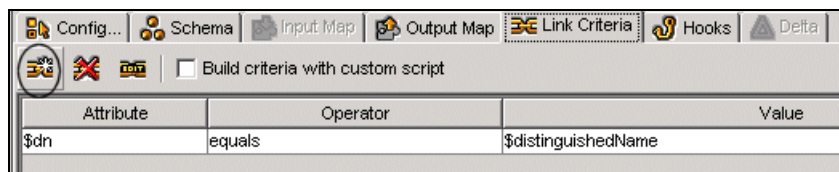
Figure 3-14   Add New Link Criteria button
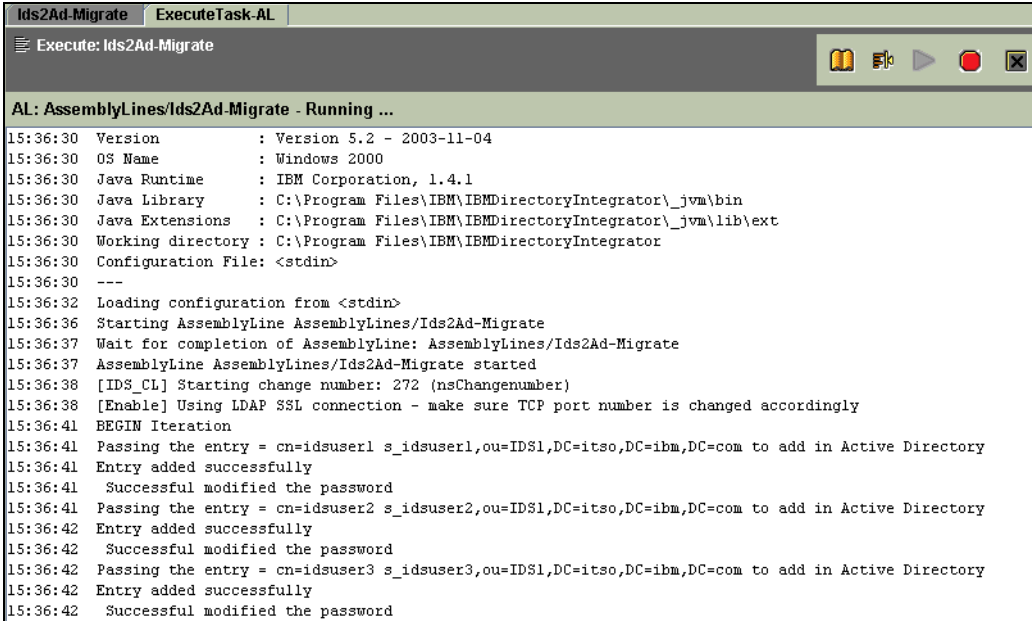
3. Click the **OK** button.

In order to enable the user in Active Directory, the userAccountControl attribute needs to be set. The attribute is set in the Output map.

The Output map contains two attributes:

1. **userAccountControl**: set to value "544**"**

2. **userpassword**: `userpassword`

This completes the configuration for migrating Directory Server users to Active Directory. On executing the AssemblyLine, *enabled* users were added in Active Directory with their password set. Figure 3-15 shows the execution log of the migration.


```
Ids2Ad-Migrate    ExecuteTask-AL

Execute: Ids2Ad-Migrate                                    [📖] [⑂] [▷] [⬤] [✕]

AL: AssemblyLines/Ids2Ad-Migrate - Running ...
15:36:30  Version          : Version 5.2 - 2003-11-04
15:36:30  OS Name          : Windows 2000
15:36:30  Java Runtime      : IBM Corporation, 1.4.1
15:36:30  Java Library      : C:\Program Files\IBM\IBMDirectoryIntegrator\_jvm\bin
15:36:30  Java Extensions   : C:\Program Files\IBM\IBMDirectoryIntegrator\_jvm\lib\ext
15:36:30  Working directory : C:\Program Files\IBM\IBMDirectoryIntegrator
15:36:30  Configuration File: <stdin>
15:36:30  ---
15:36:32  Loading configuration from <stdin>
15:36:36  Starting AssemblyLine AssemblyLines/Ids2Ad-Migrate
15:36:37  Wait for completion of AssemblyLine: AssemblyLines/Ids2Ad-Migrate
15:36:37  AssemblyLine AssemblyLines/Ids2Ad-Migrate started
15:36:38  [IDS_CL] Starting change number: 272 (nsChangenumber)
15:36:38  [Enable] Using LDAP SSL connection - make sure TCP port number is changed accordingly
15:36:41  BEGIN Iteration
15:36:41  Passing the entry = cn=idsuser1 s_idsuser1,ou=IDS1,DC=itso,DC=ibm,DC=com to add in Active Directory
15:36:41  Entry added successfully
15:36:41   Successful modified the password
15:36:41  Passing the entry = cn=idsuser2 s_idsuser2,ou=IDS1,DC=itso,DC=ibm,DC=com to add in Active Directory
15:36:42  Entry added successfully
15:36:42   Successful modified the password
15:36:42  Passing the entry = cn=idsuser3 s_idsuser3,ou=IDS1,DC=itso,DC=ibm,DC=com to add in Active Directory
15:36:42  Entry added successfully
15:36:42   Successful modified the password
```

Figure 3-15   Directory Server user migration execution log

Configuration of an AssemblyLine to migrate data from Active Directory to Directory Server

The procedure for the configuration is as follows:

1. Create a new AssemblyLine. Add an Active Directory Changelog Connector in iterator mode. Configure it. Refer to Figure 3-16 on page 64 for the configuration of the connector.
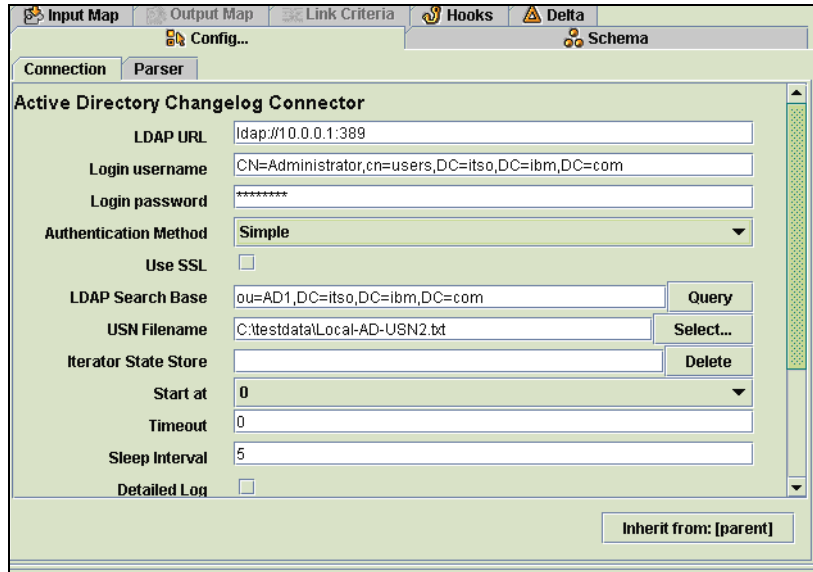
Figure 3-16   Active Directory Changelog Connector configuration

We map the Active Directory users to objectclass "person" in Directory Server and use $dn, cn, sn, objectclass and description attributes only. The user in the Active Directory has far more attributes than the attributes in objectclass person. It is difficult to map the users uniquely in the given scenario. Therefore, we are modifying the schema of objectclass person. An optional attribute uid is added to it. The sAMAccountName that is unique in Active Directory is mapped to this. Hence, the distinguished name of the entry in Directory Server is:

```
uid=<sAMAccountName from Active Directory>,ou=AD1,ou=itso,o=ibm,c=us
```

Note: Other objectclasses in Directory Server schema contain attributes similar to those of Active Directory user. Because we use the matching objectclasses, the attribute mapping is simple without need for any schema modification, which is similar to the migration case of Active Directory to Directory Server. The point here is to demonstrate the solution development in case of a constraint: certain attributes not matching.

2. In the input map, the attribute $dn from Active Directory is modified as mentioned above.

3. To add the entries to the Directory Server, an Directory Integrator LDAP Connector working in AddOnly mode is added and configured.

4. The output mapping done is as follows:

- **$dn**: $dn (modified)
- **cn**: givenName
- **description**: description
- **objectclass: v**alue "person"
- **sn**: sn
- **uid**: sAMAccountName

This completes the configuration of the AssemblyLine. On execution of the AssemblyLine, the users from Active Directory are added into Directory Server. Figure 3-17 shows the execution log of the AssemblyLine.


```
Execute: ExecuteTask-AL

AL: AssemblyLines/Ad2Ids-Migrate - Running ...
16:20:27  Version          : Version 5.2 - 2003-11-04
16:20:27  OS Name          : Windows 2000
16:20:27  Java Runtime      : IBM Corporation, 1.4.1
16:20:27  Java Library      : C:\Program Files\IBM\IBMDirectoryIntegrator\_jvm\bin
16:20:27  Java Extensions   : C:\Program Files\IBM\IBMDirectoryIntegrator\_jvm\lib\ext
16:20:27  Working directory : C:\Program Files\IBM\IBMDirectoryIntegrator
16:20:27  Configuration File: <stdin>
16:20:27  ---
16:20:29  Loading configuration from <stdin>
16:20:33  Starting AssemblyLine AssemblyLines/Ad2Ids-Migrate
16:20:34  Wait for completion of AssemblyLine: AssemblyLines/Ad2Ids-Migrate
16:20:34  AssemblyLine AssemblyLines/Ad2Ids-Migrate started
16:20:35  [AD-Change_Conn] Username specified, using Simple authentication
16:20:35  [AD-Change_Conn] Will read start USN values from file: C:\testdata\Local-AD-USN2.txt
16:20:35  BEGIN Iteration
16:21:25   Passing the entry uid=l_aduser1,ou=AD1,ou=itso,o=ibm,c=us to add into IBM Tivoli Directory Server.
16:21:25  Added entry successfully
16:22:01   Passing the entry uid=l_aduser2,ou=AD1,ou=itso,o=ibm,c=us to add into IBM Tivoli Directory Server.
16:22:01  Added entry successfully
16:22:36   Passing the entry uid=l_aduser3,ou=AD1,ou=itso,o=ibm,c=us to add into IBM Tivoli Directory Server.
16:22:36  Added entry successfully
16:23:11   Passing the entry uid=l_aduser4,ou=AD1,ou=itso,o=ibm,c=us to add into IBM Tivoli Directory Server.
16:23:11  Added entry successfully
```

Figure 3-17   Active Directory user migration execution log

> **Restriction:** Once the password for an entry in Active Directory is set, the password cannot be retrieved. Therefore, the userpassword attribute is not used for the mapping. Windows Active Directory Password Synchronizer is shipped with Directory Integrator and can retrieve the password of an entry before the password is set (added or modified). The password synchronizer is be covered in this book. For synchronization demonstration, we continue using the same attributes used for user migration.

Synchronizing data from Active Directory with Directory Server

With the current data now migrated, any user addition, modification, or deletion is done to Active Directory. For the two directories to be in synchronization, the changes should be propagated to Directory Server.

1. To retrieve the changes from Active Directory, a new AssemblyLine named Sync_Data is configured. An Active Directory changelog connector is added and configured.

   Note: For real time synchronization, the Active Directory Changelog EventHandler is used. We use the Active Directory changelog Connector with TimeOut set to zero, which has the same action as that of EventHandler.

2. To demonstrate an entry modification, we choose the field *Description*.

3. The input attribute map contains the $dn, givenName, sn, changeType, description, and sAMAccountName attributes.

4. In the Input Map, the $dn attribute is modified during mapping to map to the $dn of Directory Server.

   To propagate the three type of changes (additions, modifications, and deletions), two more AssemblyLines are configured.

1. The addition and modification can be done using a single LDAP Connector configured in Update mode. The AssemblyLine named ModNAdd2IDS is configured.

2. For deletions, the LDAP connector in delete mode is employed in the other AssemblyLine named Del4IDS.

   Depending upon the changeType, the corresponding AssemblyLines are called.

3. The scripting is done using the Default Success hook of the Active Directory Changelog Connector. The work entry formed passes to the AssemblyLines executed. Refer to Figure 3-18 on page 67.
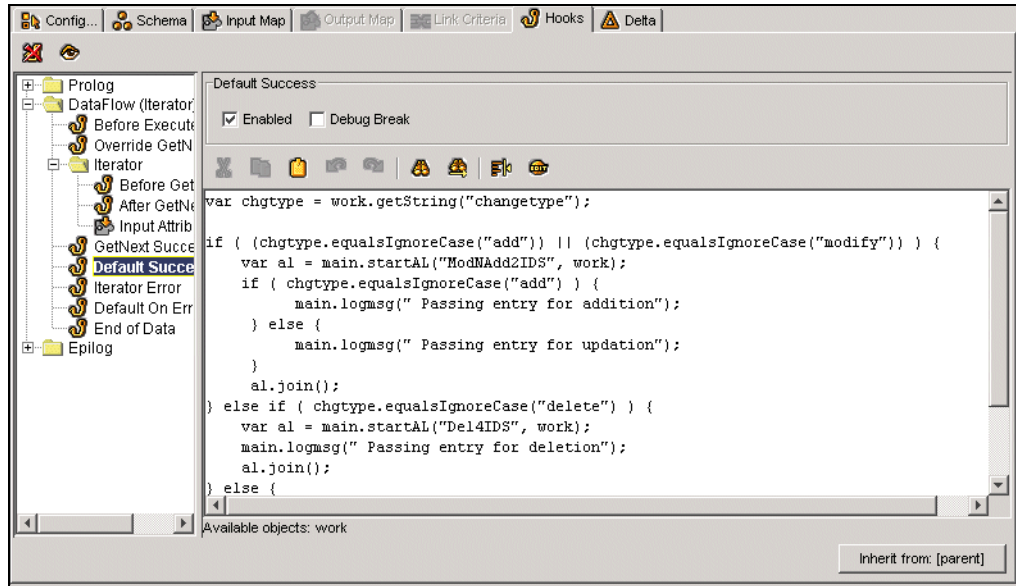
Figure 3-18   Hook containing code to call AssemblyLines

4. For the AssemblyLine ModNAdd2IDS, all the attributes are directly mapped from the work entry, except for $dn and objectclass. The $dn attribute is used for the link criteria.

5. For the AssemblyLine Del4IDS, only the $dn attribute is mapped and also used for the link criteria.

Figure 3-19 on page 68 shows the synchronization of addition type of changes.

```
Sync_data   ExecuteTask-AL
≣ Execute: Sync_data


AL: AssemblyLines/Sync_data - Running ...
16:06:58  Version          : Version 5.2 - 2003-11-04
16:06:58  OS Name          : Windows 2000
16:06:58  Java Runtime      : IBM Corporation, 1.4.1
16:06:58  Java Library      : C:\Program Files\IBM\IBMDirectoryIntegrator\_jvm\bin
16:06:58  Java Extensions   : C:\Program Files\IBM\IBMDirectoryIntegrator\_jvm\lib\ext
16:06:58  Working directory : C:\Program Files\IBM\IBMDirectoryIntegrator
16:06:58  Configuration File: <stdin>
16:06:58  ---
16:07:00  Loading configuration from <stdin>
16:07:05  Starting AssemblyLine AssemblyLines/Sync_data
16:07:05  Wait for completion of AssemblyLine: AssemblyLines/Sync_data
16:07:05  AssemblyLine AssemblyLines/Sync_data started
16:07:06  [syncAD] Will read start USN values from file: C:\testdata\Local-AD-USN2.txt
16:07:06  BEGIN Iteration
16:08:22   Passing entry for addition
16:08:22  AssemblyLine AssemblyLines/ModNAdd2IDS started
16:08:22  No iterator in AssemblyLine, will run single pass only
16:08:22  BEGIN Iteration
16:08:22  Using runtime provided entry as working entry (first pass only)
16:08:23  END Iteration
16:08:23  BEGIN Connector Statistics
16:08:23   [Modify_IDS] Lookup:1, Add:1
16:08:23  Total: Lookup:1, Add:1
16:08:23  END Connector Statistics
16:08:23  terminated successfully (0 errors)
16:08:23  AssemblyLine AssemblyLines/ModNAdd2IDS terminated successfully
16:08:23   Entry passed is uid=1_syncADuser1,ou=AD1,ou=itso,o=ibm,c=us
16:09:23   Passing entry for updating
16:09:23  AssemblyLine AssemblyLines/ModNAdd2IDS started
16:09:23  No iterator in AssemblyLine, will run single pass only
16:09:23  BEGIN Iteration
16:09:23  Using runtime provided entry as working entry (first pass only)
16:09:23  END Iteration
```

Figure 3-19   Synchronization of addition entry

Figure 3-20 on page 69 shows the synchronization of modification and deletion type of changes.

```
AL: AssemblyLines/Sync_data - Running ...
16:08:23   [Modify_IDS] Lookup:1, Add:1
16:08:23  Total: Lookup:1, Add:1
16:08:23  END Connector Statistics
16:08:23  terminated successfully (0 errors)
16:08:23  AssemblyLine AssemblyLines/ModNAdd2IDS terminated successfully
16:08:23    Entry passed is uid=1_syncADuser1,ou=AD1,ou=itso,o=ibm,c=us
16:09:23    Passing entry for updating
16:09:23  AssemblyLine AssemblyLines/ModNAdd2IDS started
16:09:23  No iterator in AssemblyLine, will run single pass only
16:09:23  BEGIN Iteration
16:09:23  Using runtime provided entry as working entry (first pass only)
16:09:23  END Iteration
16:09:23  BEGIN Connector Statistics
16:09:23    [Modify_IDS] Lookup:1, Modify:1
16:09:23  Total: Lookup:1, Modify:1
16:09:23  END Connector Statistics
16:09:23  terminated successfully (0 errors)
16:09:23  AssemblyLine AssemblyLines/ModNAdd2IDS terminated successfully
16:09:23    Entry passed is uid=1_syncADuser1,ou=AD1,ou=itso,o=ibm,c=us
16:09:58    Passing entry for deletion
16:09:58  AssemblyLine AssemblyLines/Del4IDS started
16:09:58  No iterator in AssemblyLine, will run single pass only
16:09:58  BEGIN Iteration
16:09:58  Using runtime provided entry as working entry (first pass only)
16:09:58  END Iteration
16:09:58  BEGIN Connector Statistics
16:09:58    [Del_IDS] Delete:1
16:09:58  Total: Delete:1
16:09:58  END Connector Statistics
16:09:58  terminated successfully (0 errors)
16:09:58  AssemblyLine AssemblyLines/Del4IDS terminated successfully
16:09:58    Entry passed is uid=1_syncADuser1,ou=AD1,ou=itso,o=ibm,c=us
```

Figure 3-20   Synchronization of modification and deletion of entries.

The addition, modification, and deletion type of changes in Active Directory are successfully propagated to Directory Server. Thus, one way synchronization is achieved. This completes the scenario.

Organizations do require two-way synchronization between Directory Servers. The scenario at hand is modified. If a decision is taken at this point to have two-way synchronization between Active Directory and Directory Server, the next section explains the procedure.

Two-way synchronization between Active Directory and Directory Server

The user data under the organizational units is as follows:

► `ou=AD1,DC=itso,DC=ibm,DC=com` in Active Directory
► `ou=AD1,ou=itso,o=ibm,c=us` in Directory Server

This data is synchronized. The *description* attribute can demonstrate the entry modification case.

The AssemblyLines used for synchronizing Active Directory changes with Directory Server are employed. Four new AssemblyLines are configured to synchronize Directory Server changes with Active Directory. The mapping of attributes from objectclass person (with the uid attribute added) of Directory Server to the objectclasses person, organizationalPerson, and user of Active Directory, is as follows:

- ► **$dn**: `cn + " " + sn + <parent dn>`
- ► **description**: `description`
- ► **displayName**: `cn + " " + sn`
- ► **givenName**: `cn`
- ► **sAMAccountName**: `uid`
- ► **sn**: `sn`
- ► **userPrincipalName**: `uid + "@itso.ibm.com"`

The details of configuration of the AssemblyLines follow:

AssemblyLine SyncIDS_data

1. An IBM Directory Server Changelog connector is configured in this AssemblyLine in Iterator mode.

2. All the connector attributes are mapped to the work entry in the **I**nput Map.

3. The Default Success Hook is used to detect the type of change and trigger the appropriate AssemblyLine.

4. Three new AssemblyLines, for the three cases of addition, modification, and deletion are configured.

AssemblyLine Add2AD
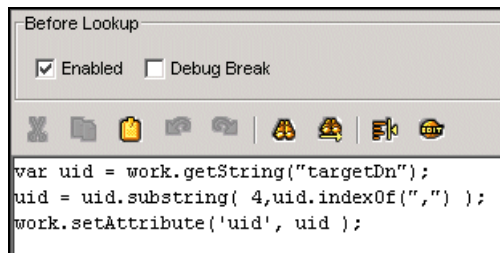
1. This AssemblyLine, with two connectors in it, is configured to synchronize the new additions made. The first connector, AD_ADD1, is employed to add the entries to Active Directory. The second connector, AD_MOD1, which works over SSL, is employed to modify the userpassword and set the userAccountControl attribute.

2. The Before Execute hook of AD_ADD1 connector contains the distinguished name building logic. Other mappings are as mentioned above.

AssemblyLine Upd-AD

1. This AssemblyLine (with two connectors in it) is configured to propagate the entry modification changes.

> **Note:** The Directory Server Changelog entry consists of distinguished name (targetDn) and only the attributes with value modification. The attributes without any value change are not present in the entry.

2. An LDAP connector IDS_Search1 in Look-up mode is employed to retrieve the attributes required for building the distinguished name. The connector looks up the Directory Server for the required entry.

3. Two attributes cn and sn are added to the Work Entry from the available connector attributes in the Input Map

4. The attribute uid is obtained by a script in the Before lookup hook. Refer to Figure 3-21.

Figure 3-21   Hook to obtain uid

5. The distinguished name is built in the Default Success hook. The LDAP connector updateAD1 working over SSL in Update mode is added to modify the entry passed.

> **Note:** The connector working in non-SSL mode works fine in this case because description is the field chosen for modification. The SSL mode enables us to modify the userpassword, if required.

6. The description and userpassword fields are mapped in the Output Map.

AssemblyLine Del-AD

1. This Directory Server AssemblyLine, with two connectors in it, is configured to propagate the deletions.

2. An LDAP connector AD_Search2 in look-up mode is employed to retrieve the attributes required for building the distinguished name. The connector looks up the Active Directory for the required entry.
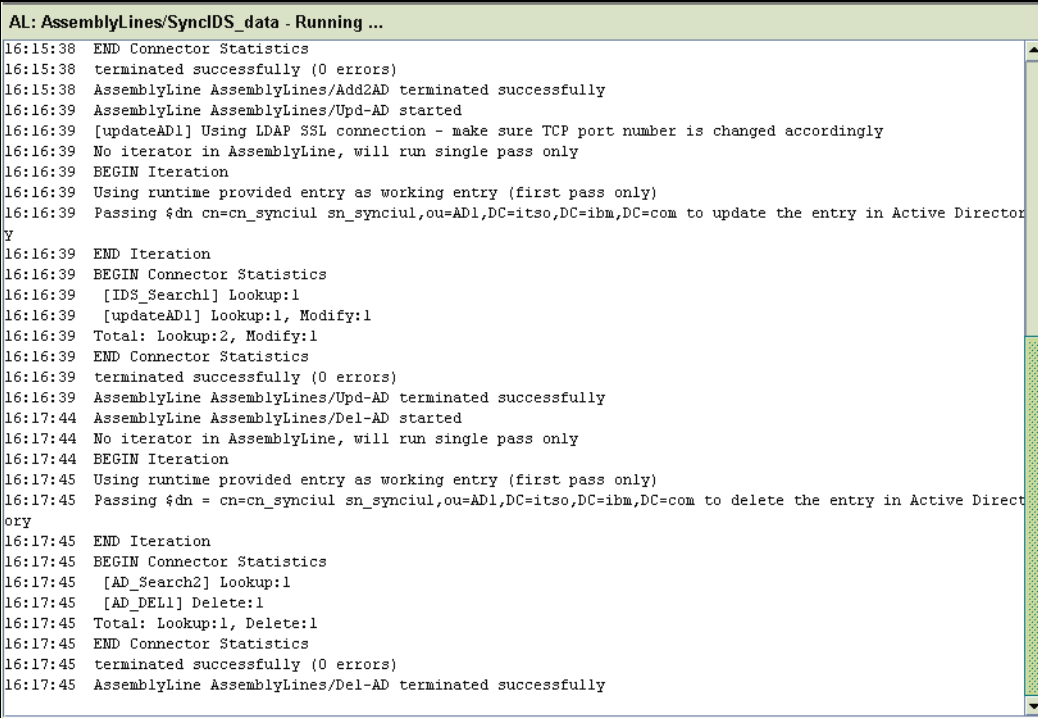
> **Note:** Since the entry is deleted, there is no entry in the Directory Server. Hence, the entry from the Active Directory is looked up.

3. Input map, configured hooks, and other details are similar to the IDS_Search1 connector.

4. The LDAP connector AD_DEL1, working in delete mode, is added to delete the entry from Active Directory.

The configuration of the AssemblyLines is now complete.

The one way synchronization from Directory Server to Active Directory is demonstrated first.

The AssemblyLine SyncIDS_data is executed (with the AssemblyLine Sync_data stopped) to propagate the changes from Directory Server to Active Directory. Figure Figure 3-22 shows the successful propagation of addition, modification, and deletion changes to Active Directory.

```
AL: AssemblyLines/SyncIDS_data - Running ...
16:15:38  END Connector Statistics
16:15:38  terminated successfully (0 errors)
16:15:38  AssemblyLine AssemblyLines/Add2AD terminated successfully
16:16:39  AssemblyLine AssemblyLines/Upd-AD started
16:16:39  [updateAD1] Using LDAP SSL connection - make sure TCP port number is changed accordingly
16:16:39  No iterator in AssemblyLine, will run single pass only
16:16:39  BEGIN Iteration
16:16:39  Using runtime provided entry as working entry (first pass only)
16:16:39  Passing $dn cn=cn_synciul sn_synciul,ou=AD1,DC=itso,DC=ibm,DC=com to update the entry in Active Director
y
16:16:39  END Iteration
16:16:39  BEGIN Connector Statistics
16:16:39   [IDS_Search1] Lookup:1
16:16:39   [updateAD1] Lookup:1, Modify:1
16:16:39  Total: Lookup:2, Modify:1
16:16:39  END Connector Statistics
16:16:39  terminated successfully (0 errors)
16:16:39  AssemblyLine AssemblyLines/Upd-AD terminated successfully
16:17:44  AssemblyLine AssemblyLines/Del-AD started
16:17:44  No iterator in AssemblyLine, will run single pass only
16:17:44  BEGIN Iteration
16:17:45  Using runtime provided entry as working entry (first pass only)
16:17:45  Passing $dn = cn=cn_synciul sn_synciul,ou=AD1,DC=itso,DC=ibm,DC=com to delete the entry in Active Direct
ory
16:17:45  END Iteration
16:17:45  BEGIN Connector Statistics
16:17:45   [AD_Search2] Lookup:1
16:17:45   [AD_DEL1] Delete:1
16:17:45  Total: Lookup:1, Delete:1
16:17:45  END Connector Statistics
16:17:45  terminated successfully (0 errors)
16:17:45  AssemblyLine AssemblyLines/Del-AD terminated successfully
```

Figure 3-22   Synchronization of Directory Server changes

Now the two-way synchronization is demonstrated.

The data in the two directories is checked for proper synchronization. The two synchronization AssemblyLines are simultaneously started. The scenario is tested in the following way:

a. A new entry is added in Active Directory. It is checked for addition in Directory Server. The entry addition triggers the Directory Server changelog Connector. The entry is picked up to add to Active Directory. The entry addition in Active Directory is not successful as the entry already exists. (Refer to figure Figure 3-23.) Then a new entry is added in Directory Server and checked in a similar way. Thus, the addition type change two-way synchronization is successful.

```
16:41:34  BEGIN Iteration
16:42:49  AssemblyLine AssemblyLines/Add2AD started
16:42:49  [AD_MOD1] Using LDAP SSL connection - make sure TCP port number is changed accordingly
16:42:53  No iterator in AssemblyLine, will run single pass only
16:42:53  BEGIN Iteration
16:42:53  Using runtime provided entry as working entry (first pass only)
16:42:53  Passing the entry = cn=newlAD1 s_newlAD1,ou=AD1,DC=itso,DC=ibm,DC=com to add in Active Directory
16:42:53  [AD_ADD1] AddOnly
javax.naming.NameAlreadyBoundException: [LDAP: error code 68 - 00000524: UpdErr: DSID-031A0AE5, problem 6005 (ENTR
Y_EXISTS), data 0
0]; remaining name 'cn=newlAD1 s_newlAD1,ou=AD1,DC=itso,DC=ibm,DC=com'
```

Figure 3-23   Two-way addition type change synchronization

b. The entry modification is similarly tested by modifying one entry each from the two directories. Figure Figure 3-24 shows the Nochange message when trying to modify an entry in Directory Server.

```
16:53:49  BEGIN Iteration
16:53:49  Using runtime provided entry as working entry (first pass only)
16:53:49  END Iteration
16:53:50  BEGIN Connector Statistics
16:53:50   [Modify_IDS] Lookup:1, Nochange:1
16:53:50  Total: Lookup:1, Nochange:1
16:53:50  END Connector Statistics
16:53:50  terminated successfully (0 errors)
16:53:50  AssemblyLine AssemblyLines/ModNAdd2IDS terminated successfully
16:53:50   Entry passed is uid=1_newlAD1,ou=AD1,ou=itso,o=ibm,c=us
```

Figure 3-24   Two-way modification type change synchronization

c. Similarly the deletion case is tested.

The two-way synchronization is thus achieved.

4

Scenario 3: Directory Server and Domino

This chapter describes the following:

▶ Importing data from the Domino Directory into the Directory Server using LDAP connectors within the Directory Integrator tool

▶ Importing data from the Directory Server into an Domino Directory using two different connectors within the Directory Integrator tool

▶ Migrating entries directly from a LDAP directory server into the Domino Directory and registering them, using the LDAP Domino Upgrade Service

4.1 Scenario 3A

A fictitious company has a Directory Server in place that contains most of the employees' details. The company uses Domino as its mail system. Therefore, it has other information in the Domino Directory that is not contained in the Directory Server. The administrators wish to migrate the data from the one in the Domino Directory to the Directory Server. This migration would create one repository that contains all of the employee information. This structure is represented in Figure 4-1.
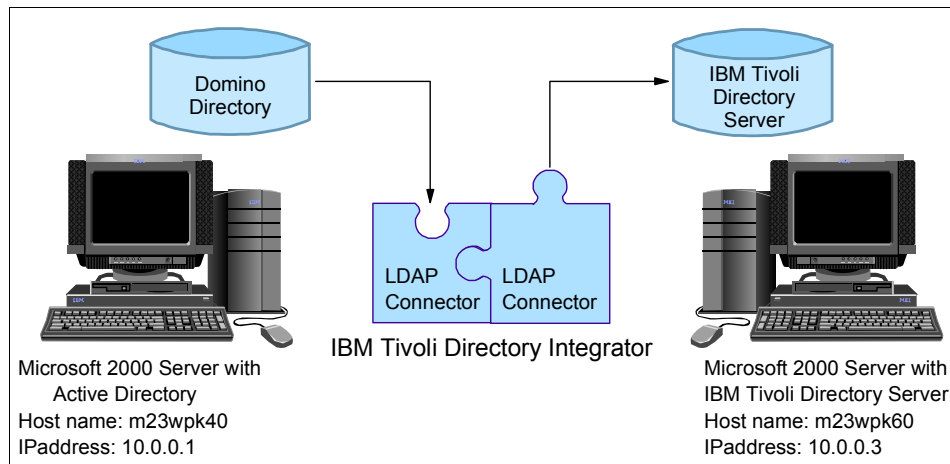
Figure 4-1   Scenario 3A

4.1.1 Prerequisites

The easiest way to get the Domino server and the Directory server communicating with one another is by using LDAP.

> **Important:** Based on our experience, LDAP is fully supported on Domino 6 or higher only. For earlier versions, we used the Internet InterORB Protocol (IIOP) connector more often than LDAP.

On the Domino server, we need to ensure that the LDAP service is running. To check whether it is running, use the Domino Administrator 6.5 client. Connect to your Domino server and then click the Server tab. Now, click the Server Tasks tab on the left-hand side. In the main window, there is a list of the tasks that are currently running on the Domino server. Scroll down the list to *LDAP Server* and make sure that its activity is set to *Listen for connect requests*. This means that the LDAP service is running correctly, as can be seen in Figure 4-2 on page 77.
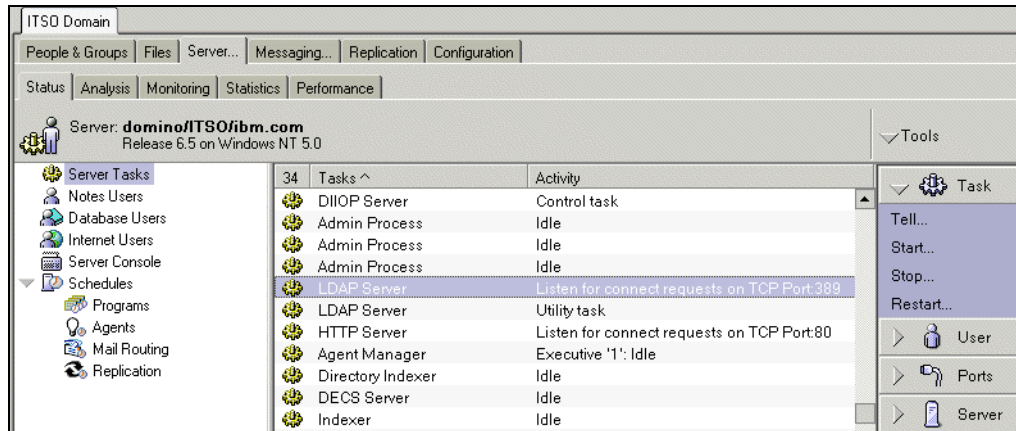
Figure 4-2   The LDAP task shown in the Domino Administrator client

If you cannot find LDAP Server on the list, you need to start the service. To do this, ensure that **Tools** is expanded on the right hand side of the client. Click **Task** → **Start ...**. In the new window that opens up, scroll down and select LDAP Server. Click the **Start Task** button. Wait a few seconds for the task to start. Then, you can click **Done** to close the window. You can now see the task LDAP Server in the list of active tasks, as shown in Figure 4-3.
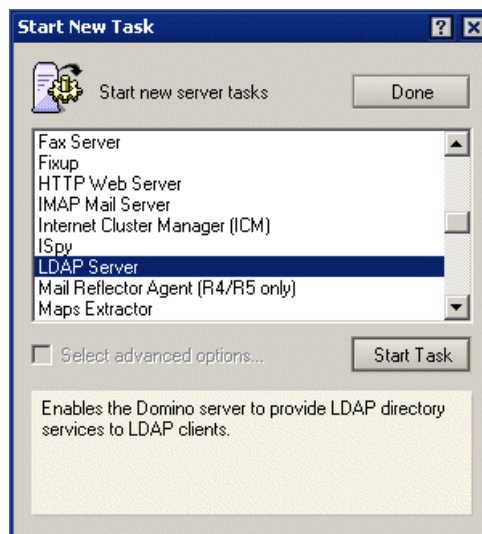
Figure 4-3   Starting the LDAP server in the Administrator client

Alternatively, you can go to the Domino server console and type:

```
load ldap
```

The LDAP service is started when you see the line, `LDAP Server: Started`.We also need to have the appropriate access to both the Domino Directory and the IBM Directory Server.

> **Note:** If you must start the LDAP service manually, it is advisable to check whether it is set to start automatically every time your Domino server restarts. To check this, open up the notes.ini file in the directory where the Domino server has been installed. Find the line that begins with ServerTasks= and check to see whether it contains the word, LDAP. If it does not contain LDAP, go to the end of that line and add **,LDAP**. The line should now look similar to this:
>
> ```
> ServerTasks=Update,Replica,Router,AMgr,AdminP,CalConn,Sched,HTTP,LDAP
> ```

4.1.2 Importing data using Directory Integrator

You can import data into Directory Server or Domino by using the Directory Integrator tool. This tool establishes a connection to each system using a built-in connector. In order to migrate the data between Domino and Directory Server successfully, we use the LDAP connector, since both Directory Server and Lotus Domino support this protocol.

More information and detailed installation instructions for Directory Integrator can be found in Chapter 3, "Scenario 2: Directory Server and Active Directory" on page 49.

4.1.3 Importing data into Directory Server

We begin by creating a new file by selecting **File** → **New...** in Directory Integrator. Enter the name of the file and click **OK**. In the left hand pane, right-click **AssemblyLines** and select **New AssemblyLine...** In the window that opens, type in the name of your assembly line, (for example, `DomIDS`) and click **OK**. The new assembly line opens in the right hand pane, as shown in Figure 4-4 on page 79. We now need to create two connectors, one to get the data from Domino and the other one to add the data to Directory Server.
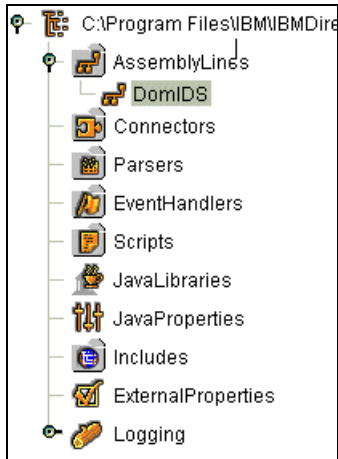
Figure 4-4   The new file with the DomIDS assembly line

Click the **Add Connector** icon (the left-most icon under the Connector window), as shown in Figure 4-5.

Figure 4-5   The Connector icons with a box around the Add new connection icon

In the Add Connector dialog box, type the name of your first connector, for example, `DomInput`. We must now select the correct connector. As mentioned earlier, we are using LDAP, so choose the system:/Connectors/ibmdi.LDAP connector from the drop down list. Ensure that the Connector Mode is set to Iterator, as shown in Figure 4-6. Click **OK** to continue.
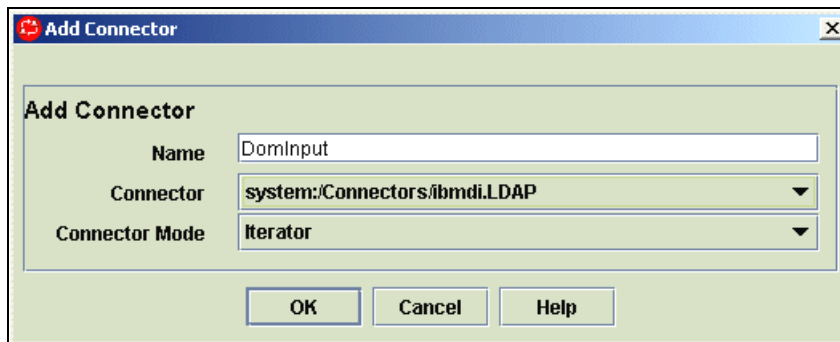
Figure 4-6   The DomInput connector

You will now see the connector's configuration panel appear in the right pane. The panel has the Config... tab open by default, as shown in Figure 4-7. These are the steps that need to be followed on the Config tab:

1. Enter the LDAP URL for the Domino Server, in the format:

   `ldap://<your Domino server's hostname or IP address>:389`

   For example, `ldap://10.0.0.2:389`

   > **Note:** To use SSL for LDAP, you would need to use a different port such as port 636, which is the default port for LDAP using SSL.

2. Next, enter the Login username and Login password. Ensure that this user has the required access to the Domino Directory (names.nsf).

3. Type in the appropriate Search Base, for example, `ou=ITSO,o=ibm.com`

4. Type in an appropriate Search Filter, for example, `objectClass=person`.

5. If you are making use of SSL, place a check mark in the box adjacent to **Use SSL.**
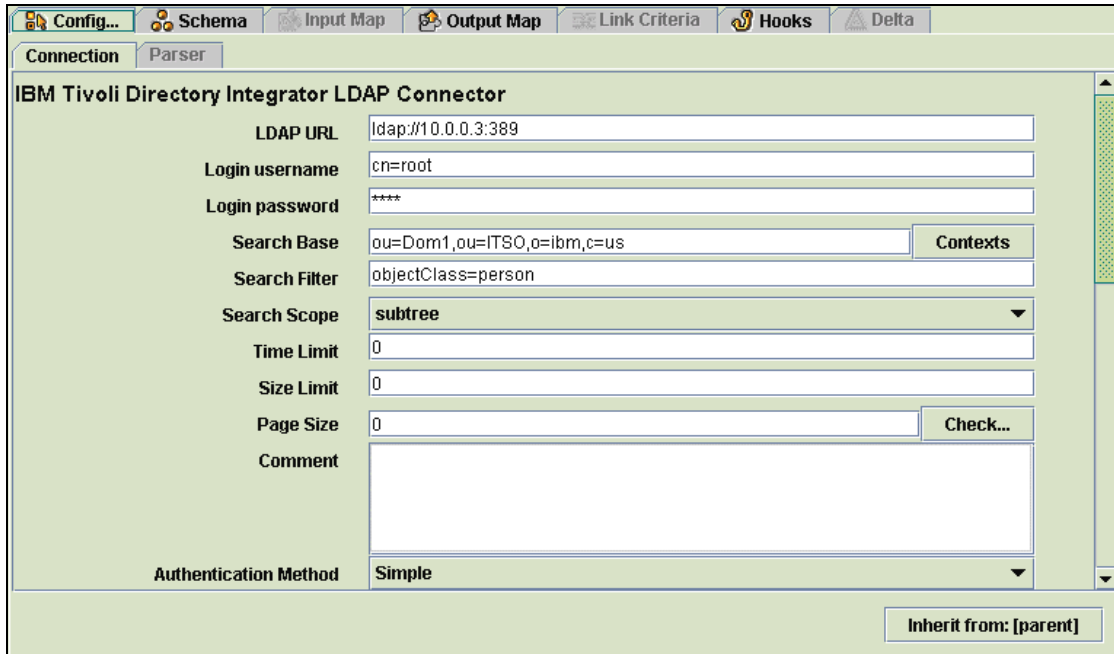

Figure 4-7   The Config settings for the DomInput connector

Now click the **Schema** tab, and do the following:

1. Click the **Connect to the data source** icon (the third icon from the left). If the connection is successful, it says Connection established adjacent to the icons. See Figure 4-8.

Figure 4-8   The Schema icons

> **Note:** If you cannot establish a connection, check all the values that you entered on the Config tab. Directory Integrator uses all of the information about the tab to establish the connection to the necessary system.

2. Click the **Read the next entry** icon (the fourth icon from the left in Figure 4-8). This will read a record in the Domino Directory and display all of the attributes associated with that record, as shown in Figure 4-9.

3. You can click the **Read the next entry** icon again to move to the next record.

Name	Java Class	Native Syntax	Sample	Input Reqd.	Output Reqd.
$dn	java.lang.String		CN=DomUser1 S_...	☐	☐
availablefordirsync	java.lang.String		1	☐	☐
checkpassword	java.lang.String		0	☐	☐
clienttype	java.lang.String		0	☐	☐
cn	java.lang.String		DomUser1 S_Dom...	☐	☐
deletentuseraccount	java.lang.String		0	☐	☐
dominocertificate	java.lang.String		03004802 684A916...	☐	☐
encryptincomingmail	java.lang.String		0	☐	☐
givenname	java.lang.String		DomUser1	☐	☐
inetpublickey	java.lang.String		30819F30 0D06092...	☐	☐
mail	java.lang.String		DomUser1@itso.ib...	☐	☐
maildomain	java.lang.String		ITSO	☐	☐
mailfile	java.lang.String		mail\domuser1	☐	☐
mailserver	java.lang.String		CN=domino,OU=IT...	☐	☐
mailsystem	java.lang.String		1	☐	☐
mailverify	java.lang.String		0	☐	☐
messagestorage	java.lang.String		1	☐	☐
netusername	java.lang.String		54caefc3ae7f874d8...	☐	☐
objectclass	java.lang.String		dominoPerson	☐	☐
passwordchangeint...	java.lang.String		0	☐	☐
passwordgraceperi...	java.lang.String		0	☐	☐

Figure 4-9   The Schema tab for the DomInput connector

Configure the Input Map tab, as shown in Figure 4-10 on page 82:

1. You are presented a list of the available attributes for this particular connector.

2. Select the attributes that you want to migrate to the Directory Server. You select attributes by clicking on the attribute in list of attributes. To highlight more than one attribute, hold the **Ctrl** key down and then click the other attributes.In this example, we select $dn, givenname, objectclass, sn, and uid.

3. Once the attributes have been selected, hold down the left-mouse key while your pointer is positioned over a selected attribute. Then, drag the selected attributes and drop them in the Work Attribute column, which is to the left of the Available Connector Attributes column.

4. If the attributes have been successfully moved to the Work Attribute column, you will see the attributes repeated under Work Entry, which is just below the Connectors.
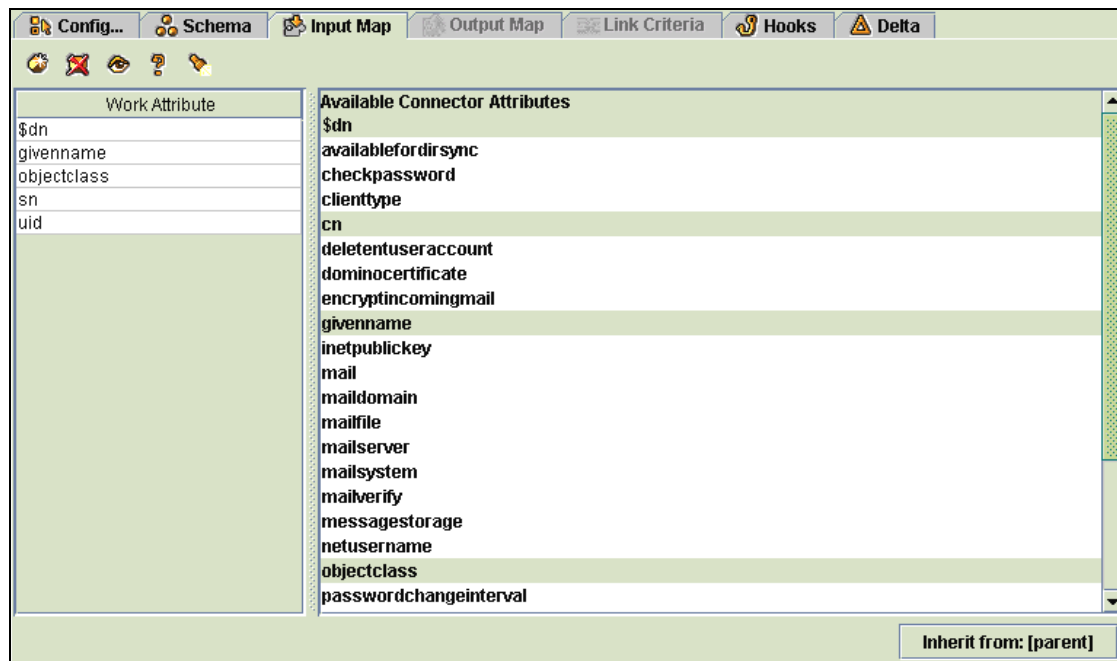
Figure 4-10   The Input map for the DomInput connector

We have completed all the required steps for the DomInput connector. Now we add a connector that adds the records from Domino into Directory Server. Go back to the Connector box. Click the **Add Connector** icon. Give the new connector a name, for example, `IDSOutput`. Again we choose the system:/Connectors/ibmdi.LDAP connector from the drop down list. Change the Connector Mode to AddOnly. Click **OK**, as shown in Figure 4-11 on page 83.
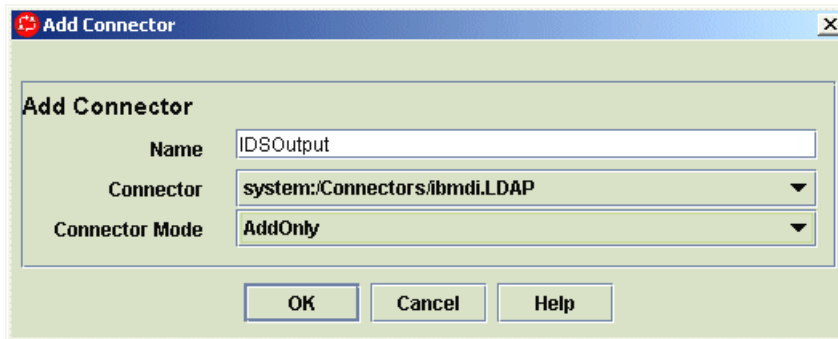
Figure 4-11   The IDSOutput connector

Notice that this connector has an Output Map tab, as shown in Figure 4-12 on page 84. This is because the mode of this connector is AddOnly, and it only adds records (to the specified system).

To set up the configuration of this connector, click the **Config...** tab.

1. Enter the LDAP URL for the IDS, such as `ldap://10.0.0.3:389`

2. Next, enter the Login username and Login password. Ensure that this user has the required access to add entries to the IDS.

   Type in the appropriate Search Base, for example,

   `ou=Dom1,ou=ITSO,o=ibm,c=us`

   Note: To check whether Directory Integrator can connect to this LDAP server, click the **Contexts** button. If it connects successfully to the LDAP system, a pop-up box will appear. In this pop-up box, you can select the relevant table that you would like to use as your search base. Find the table you wish to use and click **OK.**

3. Type in an appropriate Search Filter such as `objectClass=person`.

4. If you are making use of SSL, place a check mark in the box adjacent to **Use SSL.**

Figure 4-12   The Config settings for the IDSOutput connector

Click the **Schema** tab to connect to the Directory Server, as shown in
Figure 4-13.

1. Establish the connection to the system by clicking the **Connect to the data source** icon.

2. Click on the **Read the next entry** icon.

Figure 4-13   The Schema tab for the IDSOutput connector

Next, we need to map the Domino data to the corresponding fields in Directory Server. Click the **Output Map** tab.

1. As with the Input Map, we are presented with a list of available attributes for the IDSOutput connector.

2. We select the attributes that we want to be populated with the data from the Domino Directory. This is done by clicking the attributes in the Work Entry column, dragging them, and then dropping them in the Connector Attribute column. The attributes we chose for this example are $dn, givenname, objectclass, sn, and uid, as shown in Figure 4-14.

> **Note:** We need to ensure that we have included the required (mandatory) attributes for that object class. In this case, the attributes cn and sn are mandatory for the Person object class. Check the system's LDAP Schema to find out what the required attributes are for the object classes.
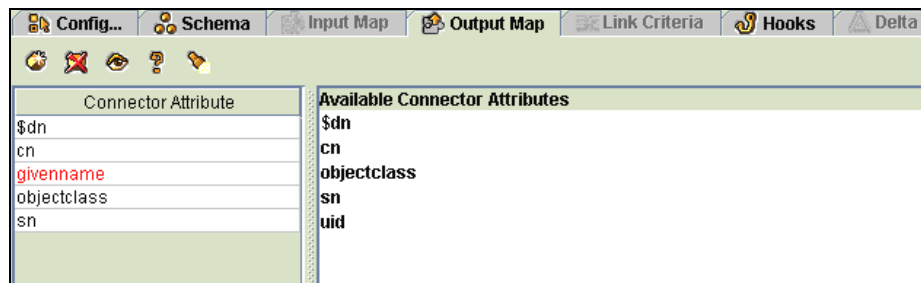
Figure 4-14   The attribute list on the Output Map for the IDSOutput connector

3. Now, we need to manipulate the Domino data so that it can be imported into Directory Server without any problems. In the Connector Attribute column, click the first entry ($dn). Notice a list of attributes in the right-hand panel, with a check mark next to $dn. This means that the connector will map the input $dn to the $dn in the Directory Server exactly. We do not want this, as the parent dn of Directory Server differs from that of the Domino Directory. Therefore, we add a script that modifies the $dn of the source (Domino) so that it will match the parent dn of the target (Directory Server). To do this, first select **$dn** from the Connector Attribute column. Then, click the box adjacent to **Advanced Mapping**. The list of attributes below it should disappear and be replaced by a blank box for us to enter our script, as shown in Figure 4-15 on page 86.
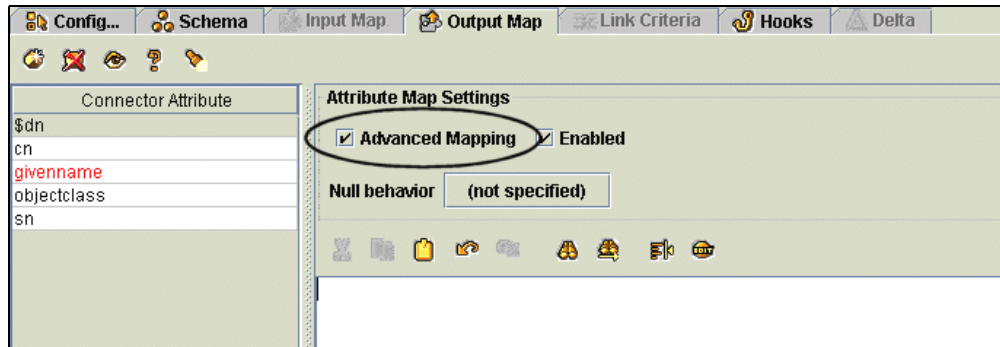
Figure 4-15   The Advanced Mapping selected with the script window opened

Enter the script below:

```
ret.value = "uid=" + work.getString("uid") + ",ou=Dom1,ou=ITSO,o=ibm,c=us"
```

This takes the uid from Domino (which is unique) and then combines it with the parent dn of the Directory Server (`ou=Dom1,ou=ITSO,o=ibm,c=us`).

> **Note:** The $dn we create needs to be unique for each record that we migrate.

4. Select the next Connector Attribute, which is **givenname**. There is no matching attribute in the Directory Server for givenname, so we rename it to the corresponding attribute, cn.

> **Note:** The Notes name of attributes and their corresponding LDAP names can be found in the Domino LDAP Schema (schema.nsf in the Data directory of the Domino Server) in the LDAP Attribute Types view.

To rename the attribute, double-click the word **givenname** in the Connector Attribute column. A cursor appears in the attribute cell. Delete the existing text and type **cn** and then press **Enter.** You now see that the new attribute (cn) is still mapped to the work entry attribute (givenname), as shown in Figure 4-16 on page 87. This is exactly what we want.
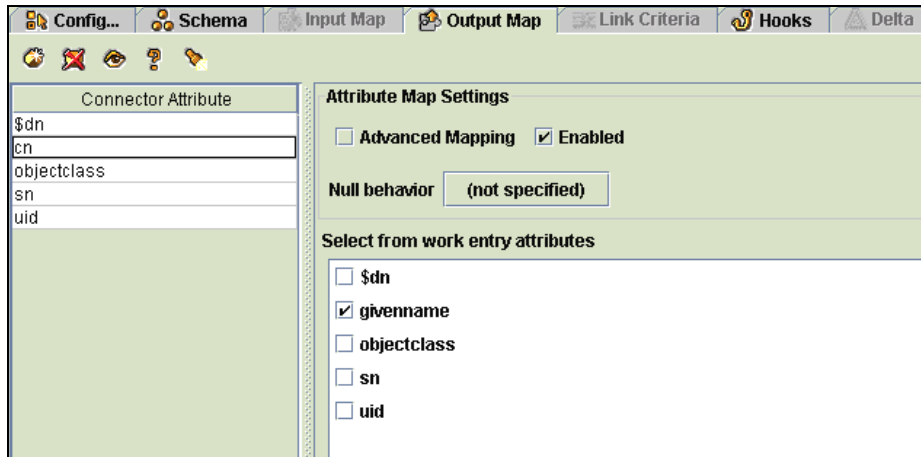
Figure 4-16   The renamed attribute, cn, mapped to givenname

5. The next attribute we need to modify is objectclass. The objectclass that comes from the Domino System contains some classes that are not recognized by Directory Server. Therefore, we limit the object to class to one common class, namely person. Click the checkbox adjacent to **Advanced Mapping**, and in the script as shown in Example 4-1.

Example 4-1   Advanced Mapping script

```
var tmpObj = system.newAttribute("objectClass");
tmpObj.addValue("person");
ret.value = tmpObj;
```

6. The next two attributes, sn and uid, can be left as is because there is no problem if they are mapped directly from the one system to the other one.

We have now completed the two connectors that we are using. If you are satisfied that your connectors are absolutely correct, you can click the **Run** icon found in the upper right hand corner of the right hand pane, shown in Figure 4-17.

Figure 4-17   The Run icon

After a few seconds, you should see Directory Integrator processing the data. The page shown in Figure 4-18 on page 88 gives you information about the work being done by Directory Integrator.

```
Execute: ExecuteTask-AL

AL: AssemblyLines/DomIDS - Stopped

15:44:30  Version           : Version 5.2 - 2003-11-04
15:44:30  OS Name           : Windows 2000
15:44:30  Java Runtime       : IBM Corporation, 1.4.1
15:44:30  Java Library       : C:\Program Files\IBM\IBMDirectoryIntegrator\_jvm\bin
15:44:30  Java Extensions    : C:\Program Files\IBM\IBMDirectoryIntegrator\_jvm\lib\ext
15:44:30  Working directory : C:\Program Files\IBM\IBMDirectoryIntegrator
15:44:30  Configuration File: <stdin>
15:44:30  ---
15:44:33  Loading configuration from <stdin>
15:44:38  Starting AssemblyLine AssemblyLines/DomIDS
15:44:38  Wait for completion of AssemblyLine: AssemblyLines/DomIDS
15:44:38  AssemblyLine AssemblyLines/DomIDS started
15:44:42  BEGIN Iteration
15:44:48  END Iteration
15:44:48  BEGIN Connector Statistics
15:44:48   [DomInput] Get:3
15:44:48   [IDSoutput] Add:3
15:44:48  Total: Get:3, Add:3
15:44:49  END Connector Statistics
15:44:49  terminated successfully (0 errors)
15:44:49  AssemblyLine AssemblyLines/DomIDS terminated successfully
15:44:49  Exit after auto-run requested
************************

Process exit code = 2
```

Figure 4-18   The ExecuteAL tab

If successful, you can go to the Directory Server and check that the entries have been added according to the configuration of your connectors.

4.2 Scenario 3B

The fictitious company wants to make the Domino Directory its central repository for user information. In order to do this, it needs to import users' records from the Directory Server. As in the previous section, we use Directory Integrator with two connectors. The major difference this time is that we use one LDAP connector to connect to the Directory Server data and one Notes connector to connect to the Domino Directory, as shown in Figure 4-19 on page 89.
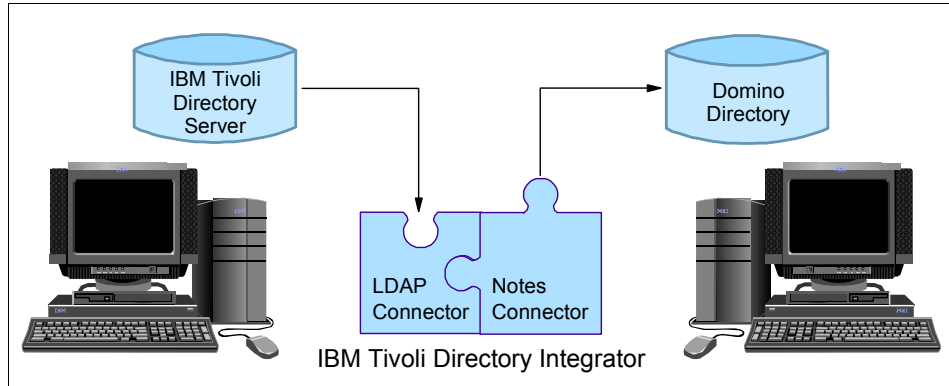
Figure 4-19   Scenario 3B

4.2.1 Importing data into the Domino Directory

We begin by creating a new assembly line, called `IDSDom`. Then, add our first connector to this assembly line by clicking the **Add Connector** icon. Give the connector a name, for example, `IDSInput`. Choose the **system:/Connectors/ibmdi.LDAP** connector from the drop down list. Ensure that the Connector Mode is set to Iterator. Click **OK** to continue. Refer to Figure 4-20.
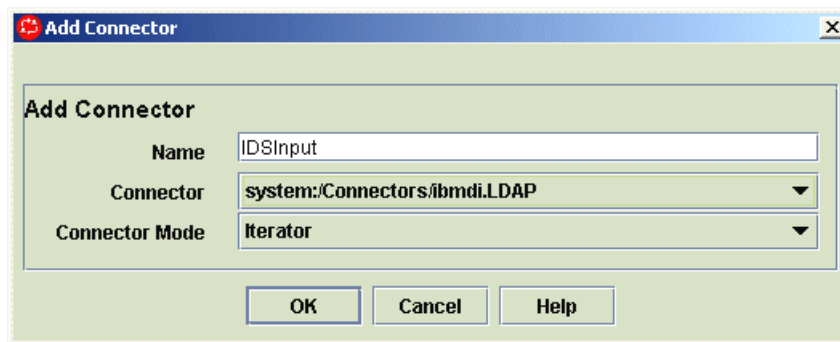
Figure 4-20   The IDSInput connector

On the Config tab, fill in the following values, as shown in Figure 4-21 on page 90:

1. Enter the LDAP URL for the Directory Server, such as `ldap://10.0.0.3:389`.

2. Enter the Login username and Login password of a user who has sufficient access rights to Directory Server.

3. Type in the appropriate Search Base, such as `ou=Dom1,ou=ITSO,o=ibm,c=us`.

4. Type in an appropriate Search Filter, for example, `objectClass=person`.

Figure 4-21   The Config settings for the IDSInput connector

On the Schema tab, ensure that you can establish a connection to Directory Server, as described in "Importing data into Directory Server" on page 78.

On the Input Map tab as shown in Figure 4-22 on page 91:

1. Select the attributes that you want to migrate to the Domino Directory. In this example, we select **$dn**, **cn**, **objectclass**, **sn**, and **uid**.

2. Drag the selected attributes and drop them in the **Work Attribute** column.
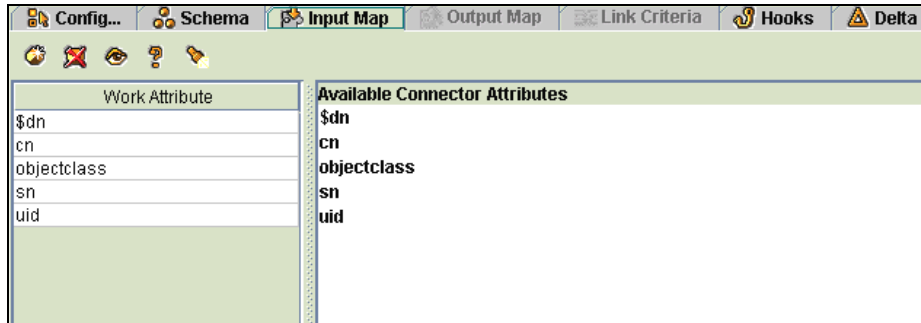
Figure 4-22   The Input Map for the IDSInput connector

Our first connector is complete. We now add the second connector that will connect to the Domino Directory and add the entries it receives from the first connector. Click the **Add Connector** again and give your connector a descriptive name, for example, DomOutput. This connector's type is system:/Connectors/ibmdi.Notes. The Connector mode must be changed to AddOnly. Refer to Figure 4-23.
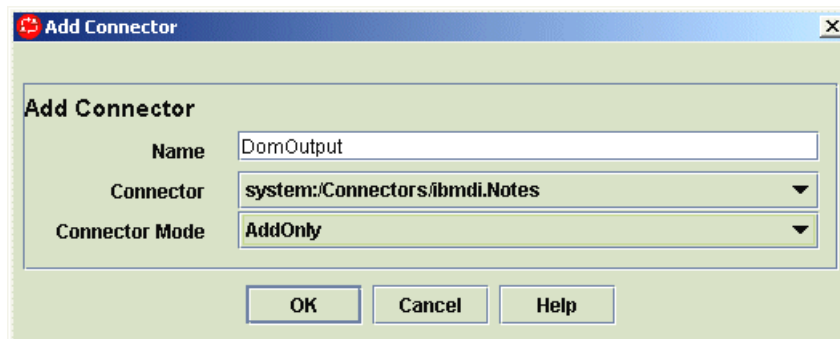
Figure 4-23   The DomOutput connector

Click **OK** and go to the Config... tab, as shown in Figure 4-25 on page 93.

1. Type the hostname or IP address of the Domino Server in the Hostname field, for example, `10.0.0.2`.

2. Type the user name and password of a person who has sufficient access to add people to the Domino Directory in the Username and Password fields, respectively.

3. Ensure that the Session Type is set to `IIOP`.

4. Type the name of the Domino server in the Server field, for example, `Domino/ITSO/ibm.com`.

5. Ensure that names.nsf is in the Database field.

> **Note:** We want names.nsf in the database field because we want to add people to the Domino Directory. You may select any Notes database that is located on that Domino server, depending on the type of data you wish to migrate.

6. Finally, select the view where you would like the records to appear in the Database View field. You can click the **Select** button adjacent to this field for a list of available views associated with that database, as shown in Figure 4-24.
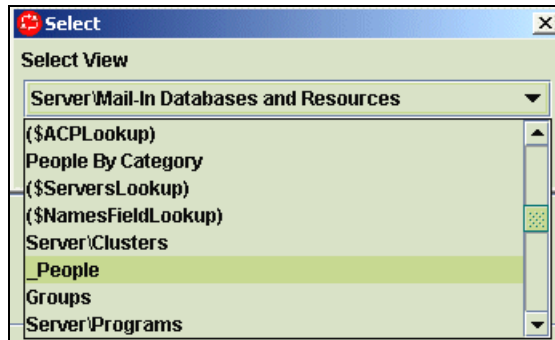
Figure 4-24   The Database View drop-down list for a Notes connector

We selected the _People view for this example.
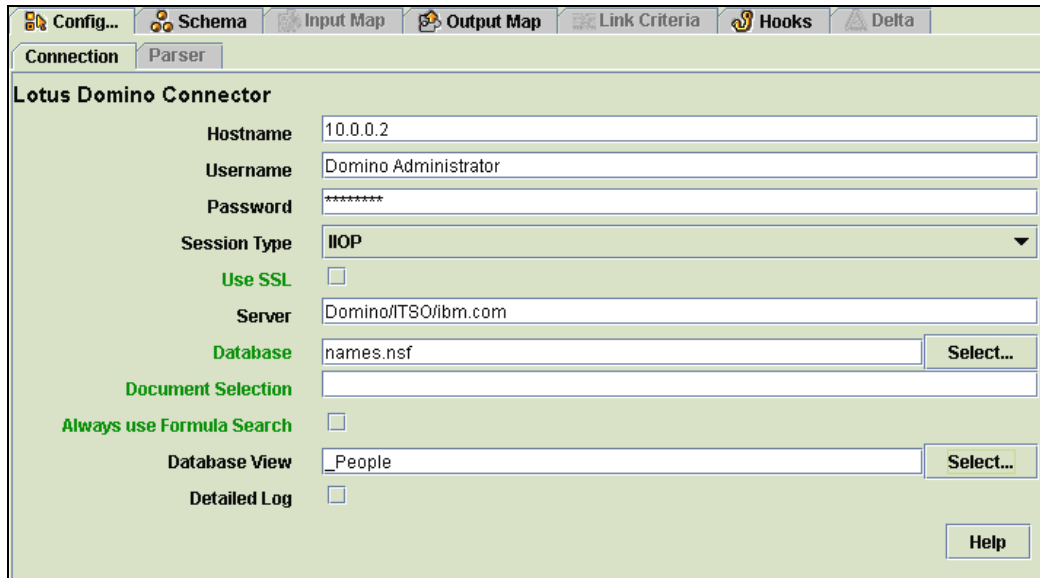
Figure 4-25   The Config settings for the DomOutput connector

Moving on to the Schema tab, you can again make sure that you can establish a connection to the data source, which is the Domino Directory in this case.

On the Output Map tab, as shown in Figure 4-26 on page 95, we need to make a number of adjustments in order for the IBM Directory Server data to be imported correctly into the Domino Directory.

1. Select all five Work Entry attributes and drag them to the **Connector Attribute** column.

2. Click **$dn** in the Connector Attribute column. Place a check mark in the box adjacent to **Advanced Mapping** in the Attribute Map Settings pane. Now, in the script box, we need to insert a script that will convert the IDS dn to the Domino dn. Type the following into the script box:

   ```
   ret.value = "CN=" + work.getString("cn") + " " + work.getString("sn") +
   ",ou=ITSO,o=ibm.com";
   ```

3. We need to rename some of the attributes so that they are be recognized by the Domino Directory. Double-click the **cn** Connector Attribute and rename it to `FirstName`.

4. Double-click the **sn** attribute and rename it to `LastName`.

5. Double-click **uid** and rename it to `ShortName`.

6. Select the **objectclass** attribute, and enable Advanced Mapping. We need to add some script to ensure that the Domino Directory's object classes are used. Use the script shown in Figure 4-2.

Example 4-2   objectclass Advanced Mapping script

```
var tmpObj = system.newAttribute("objectClass");
tmpObj.addValue("top");
tmpObj.addValue("dominoPerson");
tmpObj.addValue("inetOrgPerson");
tmpObj.addValue("organizationalPerson");
tmpObj.addValue("person");
ret.value = tmpObj;
```

7. We need to add two more Connector attributes for the migration to work successfully. Click on the **Add a new attribute to the Attribute Map** icon. This is the icon on the left directly above the Connector Attribute column. Use the drop-down list and select **FullName** and click **OK**.

8. Select the new attribute you just created (**FullName**) and then click the box adjacent to **Advanced Mapping**, in order for the script window to be displayed. Type the following script:

```
ret.value = work.getString("cn") + " " + work.getString("sn");
```

9. Add another Connector Attribute. From the drop down list, select **Type**. Click **OK**.

10. Select **Type** from the Connector Attribute column and then enable the Advanced Mapping. In the script window type the following:

```
ret.value = "Person";
```

> **Note:** This attribute is required in order to display the names of the people we are importing in the Domino Directory's People view.

The output map is depicted in Figure 4-26 on page 95.
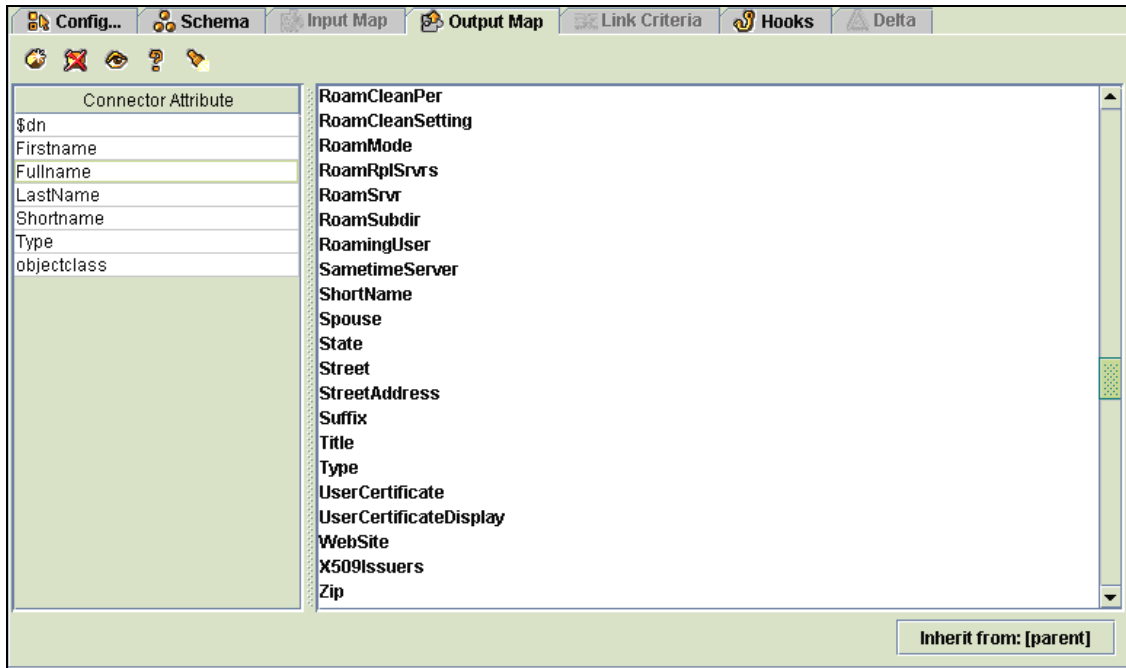
Figure 4-26   DomOutput's output map

We have now completed our two connectors and are able to run the Directory Integrator tool by clicking the **Run** icon. If it completes successfully, you are able to see the additional names that have been added from Directory Server in your Domino Directory, as shown in Figure 4-27 on page 96.
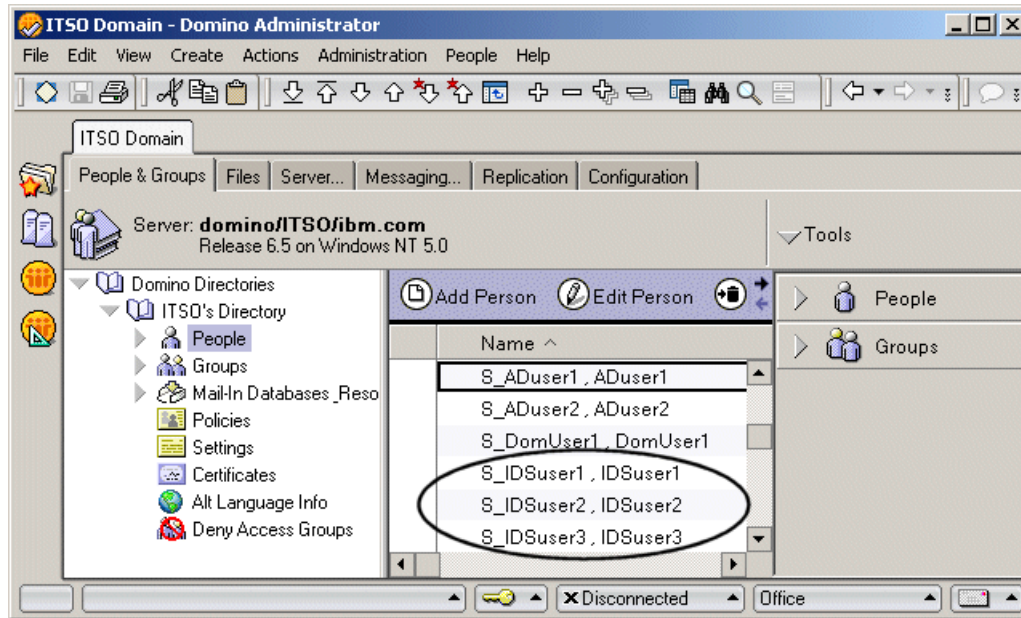
Figure 4-27   The new users that have been migrated from Directory Server

4.2.2 Synchronization between Directory Server and Domino Directory

In 4.1.3, "Importing data into Directory Server" on page 78 and 4.2, "Scenario 3B" on page 88, we discuss how to import data from one system to the other system. This is usually a once-off process, and it is not intended to be done on a regular basis. There are situations where companies would need to keep their data between two systems synchronized. This is also possible using the Directory Integrator tool. It monitors the systems for any additions, deletions, or modifications to any data in either system. An example of this can be found in Chapter 3, "Scenario 2: Directory Server and Active Directory" on page 49

4.3 Scenario 3C

In this scenario, the administrators again wish to make the Domino Directory the central repository. However, there are a number of users in the Directory Server that are not listed in the Domino Directory. The administrators wish to add these users and register them in the Domino Directory. Once registered them, each user has a Notes mail file and a Notes ID, as shown in Figure 4-28 on page 97.
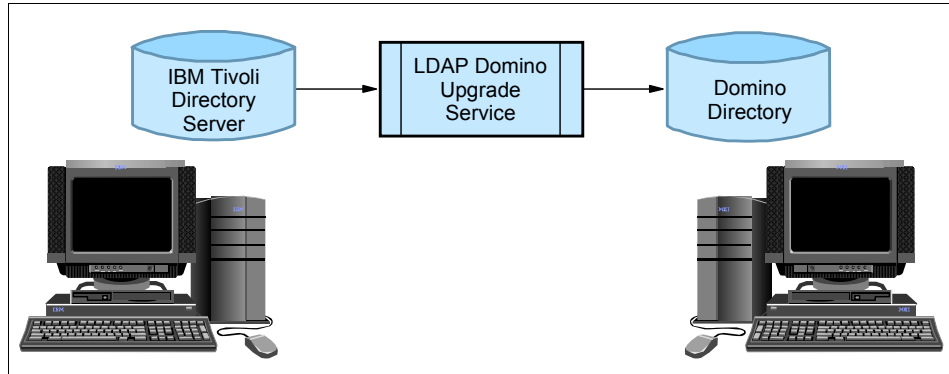
Figure 4-28   Scenario 3C

4.3.1 LDAP Domino Upgrade Service

Lotus Domino has the LDAP Domino Upgrade Service, which is used migrate users and groups from an LDAP-compliant external directory to the Domino Directory. When used, the service creates a Person document for each migrated person and a Group document for each migrated group. It is also possible to create Notes ID files and mail files for migrated users.

> **Note:** It is possible to add unregistered users (users without Notes IDs and mail files) to the Domino Directory, but these users are not be able to log into a Notes client or gain access to Domino servers and other Domino resources through Notes using Notes authentication.

When you migrate groups, they are added to the Domino Directory, but they are not registered like users.
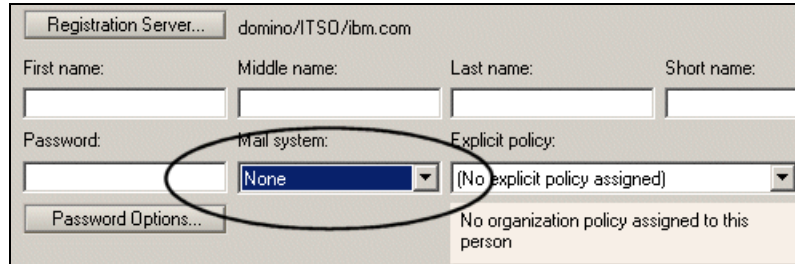
The LDAP Domino Upgrade Service offers advanced options for extending the LDAP schema in order to add person and group object classes and member attributes.

4.3.2 Migrating entries from LDAP Directory into a Domino Directory

To migrate users from a LDAP directory, we need to open the Domino Administrator client and follow these steps:

1. In the Administrator client, go to the People & Groups tab.

2. From the Tools pane, choose **People** → **Register...**

3. When prompted, choose the certifier ID and enter the password.

4. In the Basics tab of the Register Person dialog box, click **Registration Server**, and then select the Domino server that contains the Domino Directory in which you want to register the entries, for example, `Domino/ITSO/ibm.com`. Click **OK**.

5. If you are importing person entries and do not want Notes IDs and mail files created for the entries, do the following, as shown in Figure 4-29:

   a. In the Mail System field of the Basics pane, select **None**.

Figure 4-29   Selecting None to ensure no mail files are created for the users

   b. Deselect the option **Create a Notes ID for this person**. Select a certifier from the Certifier Name list to use for the name.

6. Click **Migrate people...** A People and Groups Migration dialog box will now open.

7. In the People and Groups Migration dialog box, select **LDAP** in the Foreign Directory Source field, as shown in Figure 4-30 on page 99.
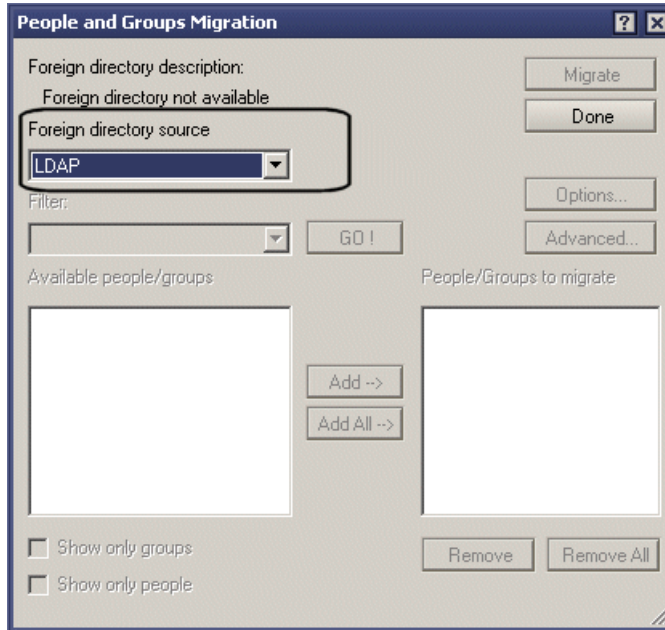
Figure 4-30   The People and Groups Migration dialog box with LDAP selected

A LDAP Domino Upgrade Service dialog box now opens.

8. In the LDAP Directory Server Properties section, enter the required information, as shown in Figure 4-31 on page 100:

   a. the LDAP Hostname, for example, `10.0.0.3`

   b. LDAP port: `389` (`636` if using the default SSL port)

   c. Base DN for search, where we use `ou=Dom1,ou=itso,o=ibm,c=us`
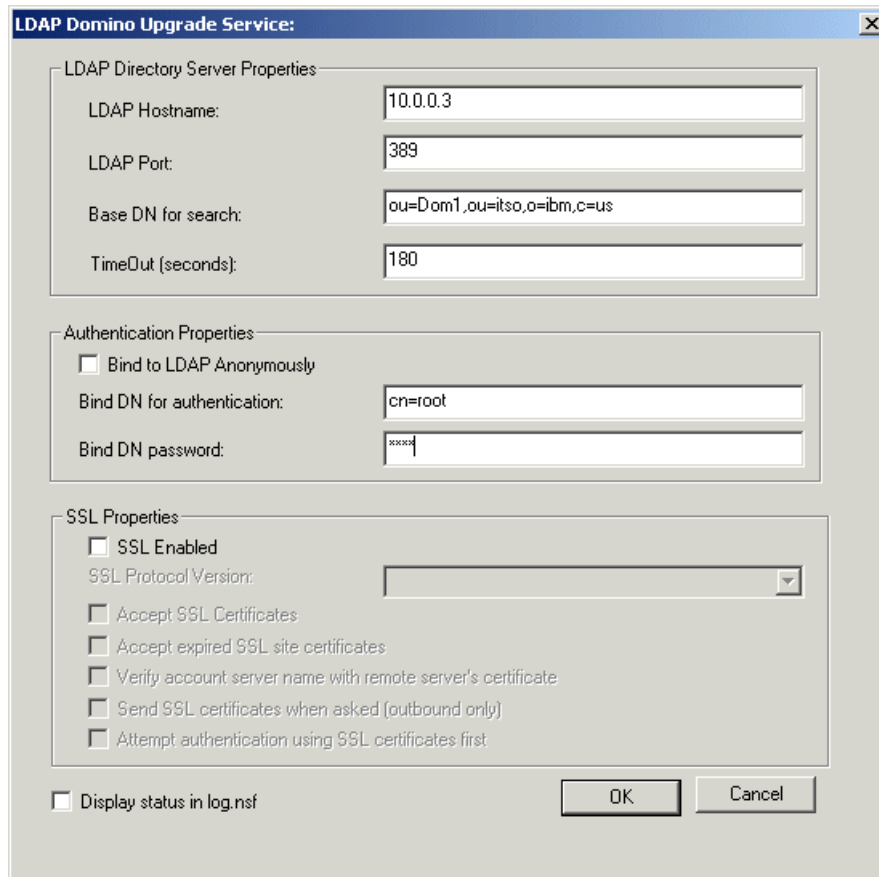
Figure 4-31   The LDAP Domino Upgrade Service dialog box

9. In the Authentication Properties section, enter the following:

    a. the Bind DN for authentication, for example, `cn=root`

    b. the Bind DN's password

10. Select whether you want to use SSL and fill in the required information

11. Click **OK**

12. You are taken back to the People and Groups Migration dialog box. Click **GO!** to see all the available people and groups.

13. Select the people/groups you wish to be registered in the Domino Directory and click **Add** →. Refer to Figure 4-32 on page 101.
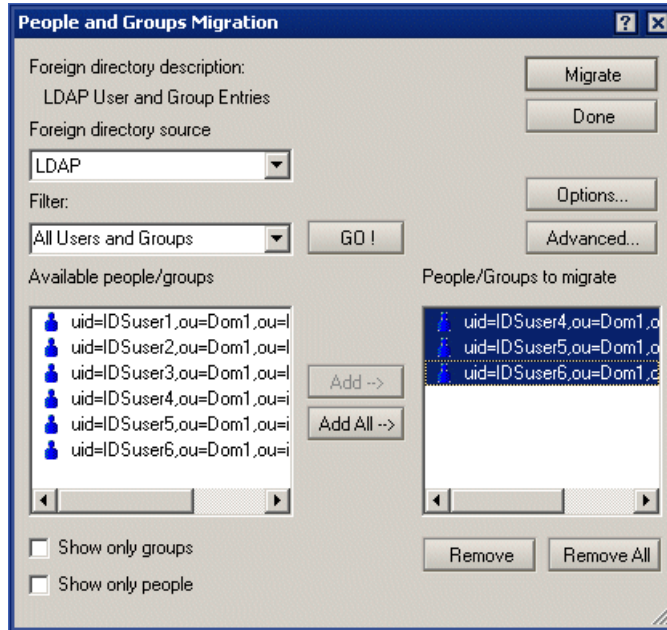
Figure 4-32   The People and Groups Migration dialog box

14. At this point, you can select different options for the users that you will be registering. To do this, click **Options...** This opens the Migration options dialog box, as shown in Figure 4-33.
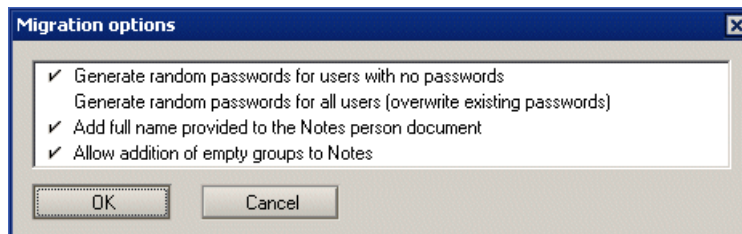
Figure 4-33   The Migration options dialog box

Select the options that you require and click **OK** to continue.

15. On the People and Groups Migration box, click **Migrate** to add the selected names to the Notes registration queue. If successful, you will receive a pop-up window informing you that the migration is successful. This is shown in Figure 4-34 on page 102.
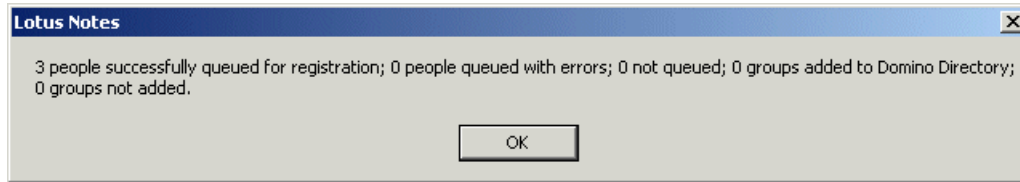
Figure 4-34   A pop-up box informing you that the people have been queued successfully

16. You are now back at the Domino Administrator Register Person screen, as shown in Figure 4-35. The people that you previously selected from the LDAP Directory server are visible in the registration queue. You notice that the fields have been filled in with information obtained from the Directory server. A random password is also created for each user. (See Migration options in Figure 4-33 on page 101.)
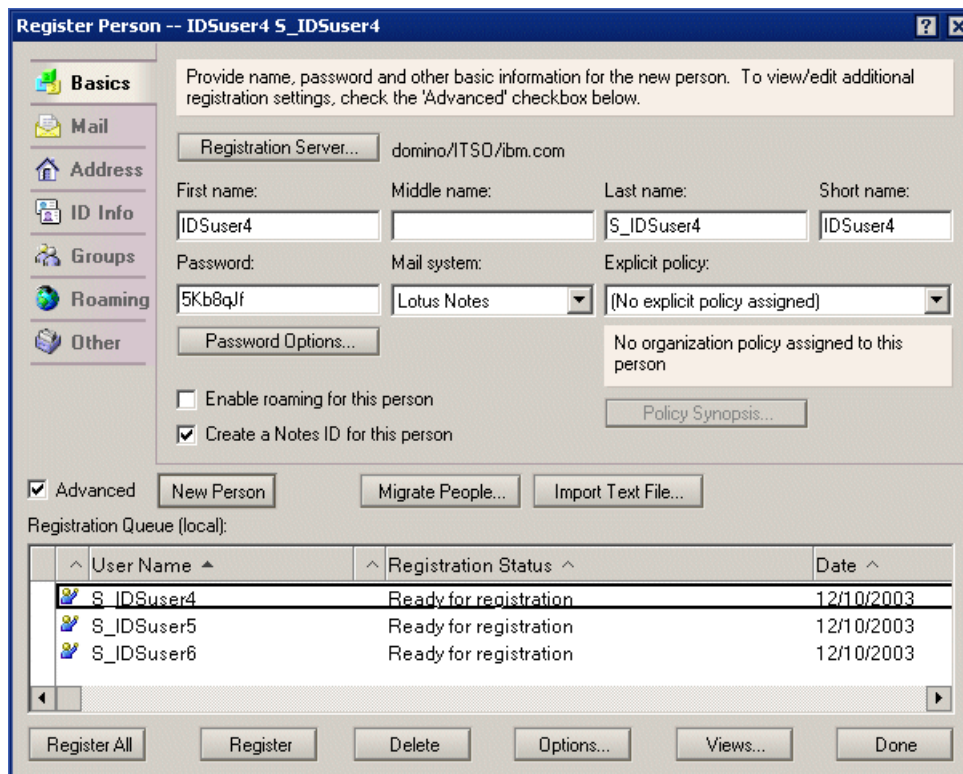
Figure 4-35   Users from the LDAP Directory server in the Registration Queue

17. Click **Register All** to register the users in the Registration Queue. This process creates a new Person record, Notes ID, and Notes mail file for each

user in the queue. The new users can be seen in the People view of the People & Groups tab in the Domino Administrator client, which is shown in Figure 4-36.
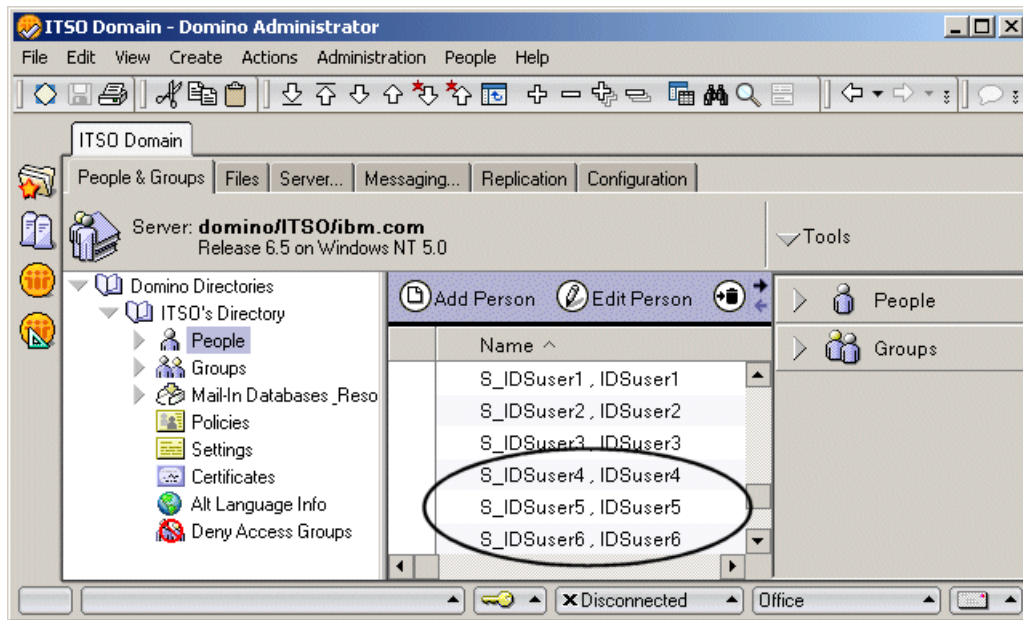
Figure 4-36   The new users from the LDAP Directory server

5

Scenario 4: Integrating data from three different sources

In the previous chapters, we described using Directory Integrator to import data from one source. This chapter describes how to get specific data from two directories, namely, an IBM Tivoli Directory and a Domino Directory, using information from a flat file for the lookups. This information will then be combined and added to an Active Directory.

5.1 Scenario 4

A fictitious company has information about its employees in various different directories. It uses Active Directory as its main directory. It also has Directory Server, which stores the Human Resources information for the employees, and a Domino Directory containing the e-mail addresses of the employees. There are a number of individuals who have not yet been added to the Active Directory. To add these individuals to the Active Directory, the administrators need to get (1) the required information from the Directory Server and (2) their e-mail addresses from the Lotus Domino Directory. Then, this information needs to be added to the Active Directory. The user IDs of these individuals are supplied in a flat file, as shown in Figure 5-1.
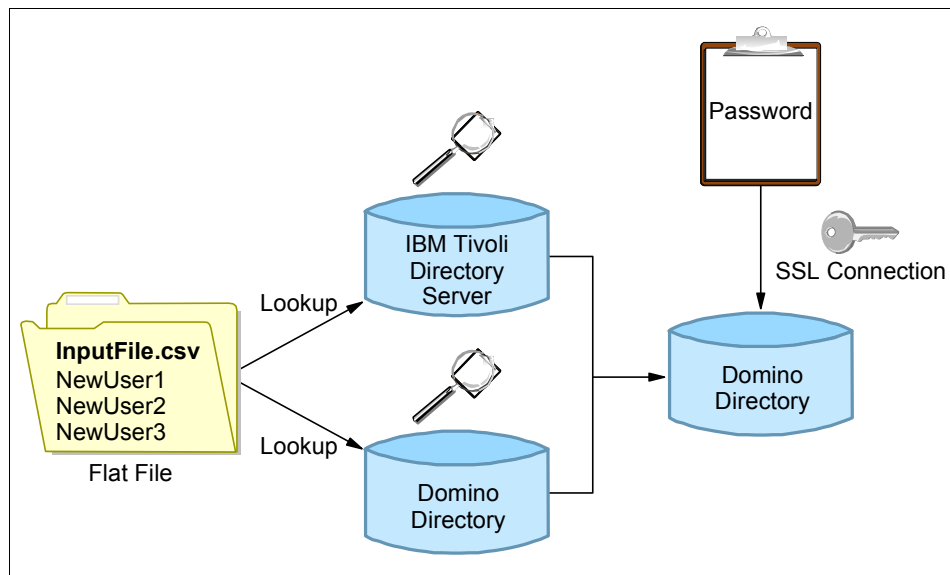
Figure 5-1   Scenario 4

We use Directory Integrator to accomplish this task because it has been designed to integrate data from a variety of sources. To accomplish this integration, we use five different connectors in the tool. The first connector connects to the flat file and iterates through each of the entries contained therein. Using the information from the flat file, the following two connectors connect to their respective directories and gather the required information. This information is then added to Active Directory by means of an AddOnly connector. The final connector connects to Active Directory using a SSL connection. Then, it sets the password for each user.

For this scenario, we create a flat file called InputFile.csv that contains the unique identifiers of the records that we wish to add to Active Directory. In this case, it contains the text shown in Example 5-1.

Example 5-1   Scenario 4 input file

```
NewUser1;
NewUser2;
NewUser3;
```

The Directory Server contains three entries, one for each person listed in the flat file. Each record contains the cn, sn, uid, description and telephone number for that particular person. These records are found in `ou=IDS1,ou=itso,o=ibm,c=us`.

In the Domino Directory, we have another three entries corresponding to the records in the Directory Server. However, the Domino Directory contains additional information that is not present in the LDAP server, such as mail and givenname.

5.2 Building the Assembly Line

Launch Directory Integrator if it is not already running. Select **File** → **New** from the menu, and name the file appropriately (for example, `multiple.xml`). Right-click the **AssemblyLines** icon and select **New AssemblyLine**. Give this assembly line a name such as `MultipleSources`, and click **OK**.

5.2.1 The File System Connector

We add our first connector by clicking on the **Add Connector** icon. Give this connector a descriptive name such as `FlatFileConn`. Change the Connector type to `system:/Connectors/ibmdi.FileSystem` and ensure that the Connector Mode is set to `Iterator`, as shown in Figure 5-2 on page 108.
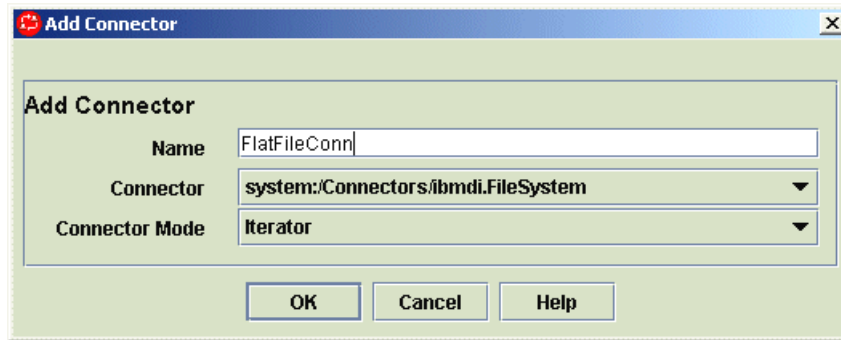
Figure 5-2   The FlatFileConn connector settings

In the Config tab, we need to insert the File Path. In this case it is InputFile.csv. Since we are using a File System connector, we have to select the relevant parser that the connector uses to interpret the data from the file. Click the **Inheritance Configuration** icon that is located in the upper left-hand corner of the configuration panel. It is shown in Figure 5-3.
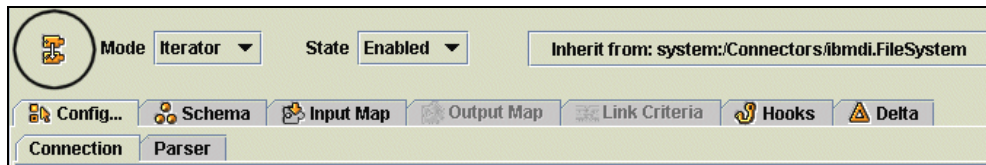
Figure 5-3   The Inheritance Configuration icon shown circled

This opens up the Configure Inheritance dialog box. In this box, select **system:/Parsers/ibmdi.CSV** as our Parser. Refer to Figure 5-4 on page 109.
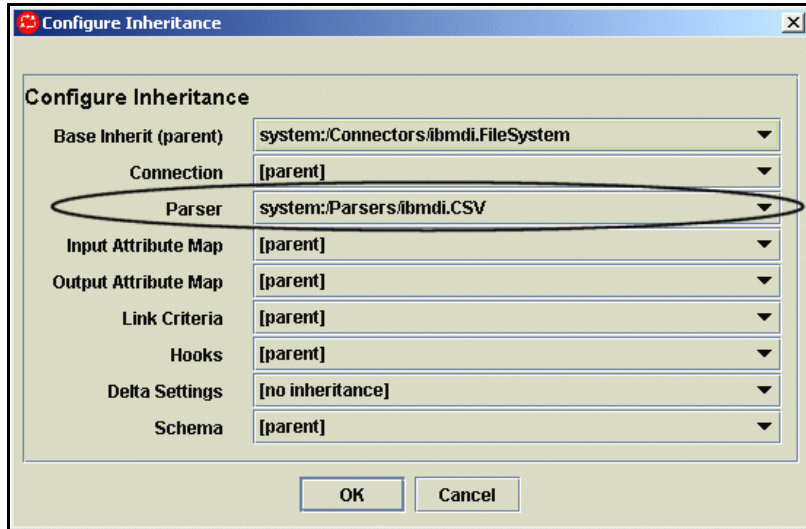
Figure 5-4   The Configure Inheritance dialog box with parser highlighted

> **Note:** We selected the CSV parser because our flat file is saved in CSV format. If your flat file is saved in a different format, you should choose the corresponding parser.

Click **OK** to return to the connector's configuration panel.

Now, click the **Parser** tab (adjacent to the Connection tab). We need to specify what the field separator is (if we have more than one field) and what field names are contained in the flat file. In our scenario, we have only one field, so we enter `uid` in the Field Names text box.

Next, click the **Schema** tab and ensure that we can establish a session to the flat file. If this is successful, click the **Input Map** tab.

We have only one available connector attribute, **uid**, which we drag to the Work Attribute Column. The File Connector is now complete.

5.2.2 The Lookup Connectors

We use two look-up connectors to connect to the two directories and gather the required data from them.

The two connectors, as shown in Table 5-1 on page 110, should be configured in the following way:

Table 5-1   The configuration of the Lookup connectors

Name	Connector	Connector Mode
IDSInput	`system:/Connectors/ibmd i.LDAP`	`Lookup`
DomInput	`system:/Connectors/ibmd i.LDAP`	`Lookup`

IDSInput connector

Click the **Config Tab** to do the following:

1. Enter the LDAP URL for the IDS. For example, `enter ldap://10.0.0.3:389`.

2. Enter the Login username and Login password of a user who has sufficient access rights to Directory Server.

3. Type in the appropriate Search Base, such as `ou=IDS1,ou=ITSO,o=ibm,c=us`.

4. Type in an appropriate Search Filter, such as `uid=*`.

Click the **Schema** tab to establish a connection to the server.

Click the **Input Map** tab and

1. Select the attributes that you want to migrate to the Domino Directory. In this example, we select **description**, **sn**, and **telephonenumber**.

2. Drag the selected attributes and drop them in the **Work Attribute** column.

DomInput connector

Click the **Config Tab** to do the following:

1. Enter the LDAP URL for the Domino Server, such as

   `ldap://10.0.0.2:389.`

2. Next, enter the Login username and Login password. Ensure that this user has the required access to the Domino Directory (names.nsf).

3. Type in the appropriate Search Base, for example,

   `ou=ITSO,o=ibm.com.`

4. Type in an appropriate Search Filter, for example, `uid=*`.

Click the **Schema** tab, to establish a connection to the server.

Click **Input Map** tab and

1. Select the attributes that you want to migrate to the Domino Directory. In this example, we select **cn**, **givenname**, and **mail**.

2. Drag the selected attributes and drop them in the **Work Attribute** column.

Our Look-up connectors are now complete.

5.2.3 The AddOnly connector

We now to need to create a connector that takes the information from the two lookup connectors and adds this information to our Active Directory. We do this by creating a AddOnly connector, which we call ADOutput. This is also a `system:/Connectors/ibmdi.LDAP` connector, and we use the AddOnly connector mode.

Click the **Config Tab**, to enter the required information:

1. Enter the LDAP URL for the Active Directory Server, such as

   `ldap://10.0.0.1:389`.

2. Next, enter the Login username and Login password. Ensure that this user has the required access to the Active Directory.

3. Type in the appropriate Search Base, such as
   `ou=IDS1,DC=itso,DC=ibm,DC=com`.

4. Type in an appropriate Search Filter, such as `cn=*`.

Click the **Schema** tab to establish a connection to the server.

Next we need to map the data from the two sources to the corresponding fields in Active Directory. Click the **Output Map** tab.

1. Select all the attributes that are listed in the Work Entry column and drag them to the **Connector Attribute** column. Refer to Figure 5-5.

Work Entry	
Name	Source
uid	FlatFileConn
description	IDSInput
sn	IDSInput
telephonenu...	IDSInput
cn	DomInput
givenname	DomInput
mail	DomInput

Figure 5-5   The Work Entry column for the MultipleSources assembly line

2. We need to rename some of the attributes. Double-click **uid** in the Connector attribute column and rename it to `sAMAccountName`.

3. Similarly rename cn to `distinguishedName`. Enable **Advanced mapping** and type the following script in the script area:

```
ret.value="cn=" + work.getString("cn") + ",ou=IDS1,DC=itso,DC=ibm,DC=com";
```

4. We need to add a few other attributes to the Connector Attribute column. Click the **Add a new attribute to the Attribute Map** icon and either select **$dn** from the drop-down list or type $dn. Enable **Advanced Mapping** using the process described in "Scenario 3A" on page 76 and type the script shown in Example 5-2:

Example 5-2   Add $dn attribute

```
var distinguishedName = "cn=" + work.getString("cn") +
",ou=IDS1,DC=itso,DC=ibm,DC=com";
work.setAttribute("distinguishedName",distinguishedName);
ret.value = distinguishedName;
```

5. The next attribute we need to add is objectclass. Again, enable **Advanced Mapping** and enter the script:

Example 5-3   Add objectclass attribute

```
var ADoc = system.newAttribute("objectClass");
ADoc.addValue("top");
ADoc.addValue("person");
ADoc.addValue("organizationalPerson");
ADoc.addValue("user");
ret.value = ADoc;
```

6. The last attribute that needs to be added is userPrincipalName. This attribute can be mapped directly to *mail*. Do this by placing a check mark in the box adjacent to mail in **Select from work entry attributes** window.

The Output Map tab should look similar to Figure 5-6 on page 113.
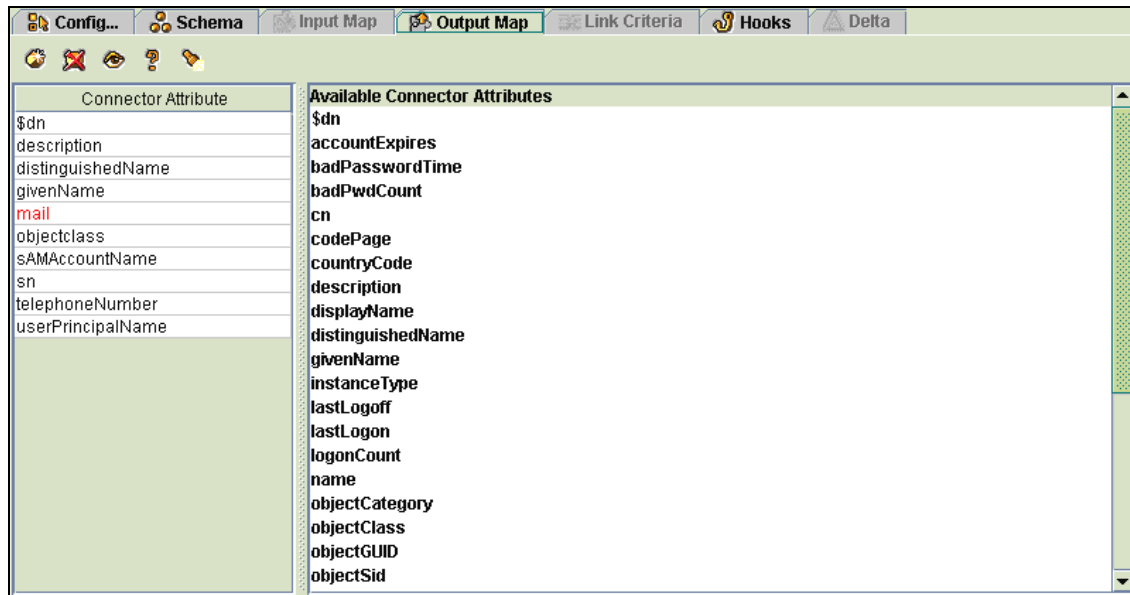
Figure 5-6   The Output Map for the ADOutput connector

Our AddOnly connector is now complete.

5.2.4 The Update connector

Our final connector connects to Active Directory via the SSL port.

It then updates the records that have been added with password information. This ensures that the new users in Active Directory are enabled and can log in anywhere in the Active Directory domain.

We add a new connector, which we name ADUserEnable. Again we use the `system:/Connectors/ibmdi.LDAP`, but this time the Connector Mode is set to Update.

The Config tab is exactly the same as the ADOutput connector, except for one important difference.

1. This time we are connecting to the SSL port of the Active Directory, and therefore the LDAP URL port is different. The entry should be:

   `ldap://10.0.0.1:636`

2. We need to scroll down the Connection page and place a check mark in the **Use SSL** check box.

Click the **Output Map** tab to add two attributes.

1. Click the **Add a new attribute to the Attribute Map** icon, and enter `userAccountControl` and click **OK**. Remove the check from the Add checkbox next to the new attribute, ensuring that only Mod is checked. Enable **Advanced Mapping** and type the following script in the script window:

   ```
   ret.value = "544";
   ```

2. Add the second attribute and name it `userPassword`. Again remove the check mark from the Add checkbox and ensure that only Mod is checked. Enable **Advanced mapping** and add this script:

   ```
   ret.value = "passw0rd";
   ```

   This sets the password to `password` for each new user added. These users would obviously need to change their password as soon as possible.

Click the **Link Criteria** tab.

1. Click the **Add new Link Criteria** icon.

2. Type $dn in the Connector Attribute field.

3. Ensure that the Operator is set to `equals`.

4. Enter $distinguishedName in the Value field.

Refer to Figure 5-7.
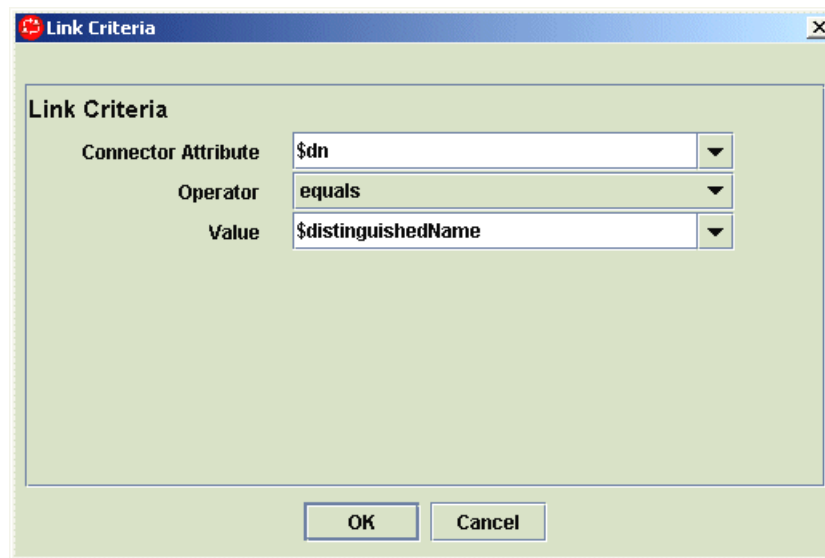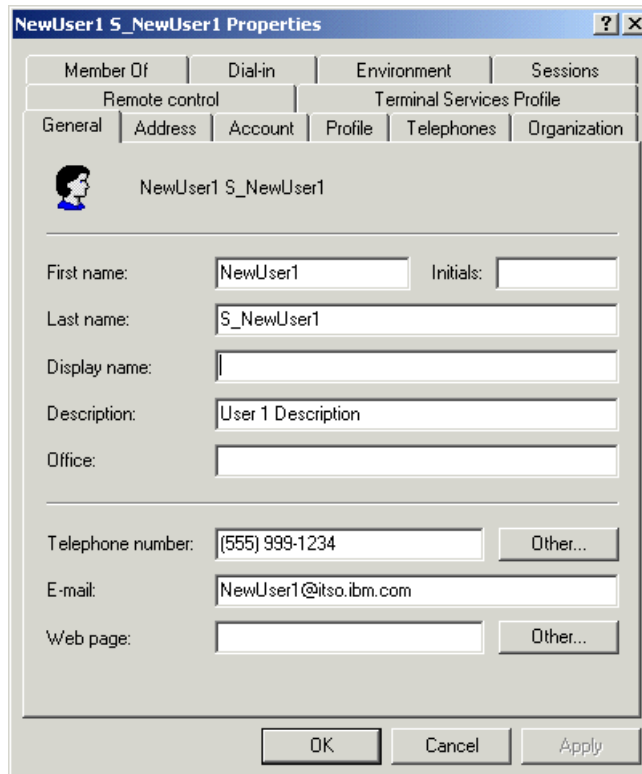
Figure 5-7   The Link Criteria for the ADUserEnable connector

5.3 The result

We have now completed all the required connectors. We now click the **Run** icon in the top right-hand side of the configuration panel and watch as the Data Integrator starts processing our assembly line. If there are no errors, you see the new users added to the Active Directory. When you look at the properties of these users, you see that it contains data from both the IBM Tivoli Directory and the Domino Directory, as shown in Figure 5-8.

Figure 5-8   The new users created in Active Directory

These users also can log on to the Active Directory domain with their usernames and passwords.
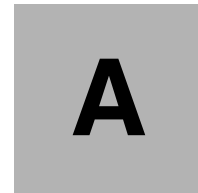
Configurations

This appendix provides information relating to the Directory Server, the Domino server, the Active Directory, and the specific configurations used in this book.

Directory Server

Operating System
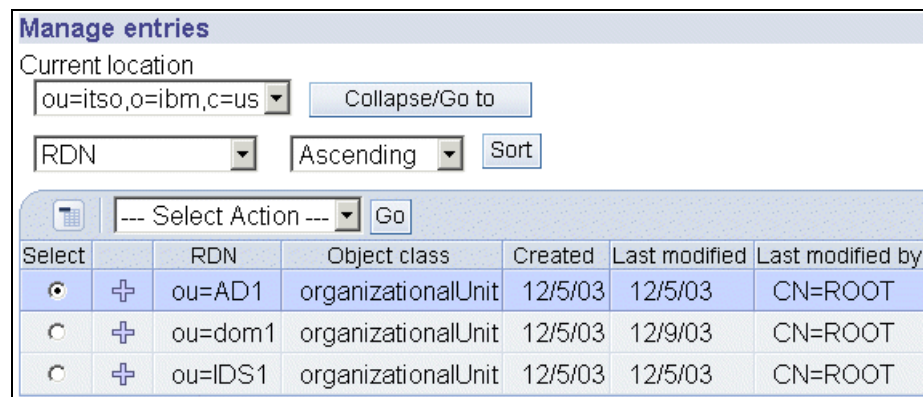
For this book, we install Directory Server 5.2 on a Microsoft Windows 2000 Server (with Service Pack 4).

Directory Server setup

We are using the suffix dn, `o=ibm,c=us`. We also used the organizational unit, `ou=itso`. Under `ou=itso,o=ibm,c=us`, we have created three further organizational units, one for each system that is used in this book. These organization units are as follows

► **Active Directory data**: `ou=IDS1`
► **Directory Server data**: `ou=AD1`
► **Lotus Domino data**: `ou=Dom1`

These settings are depicted in Figure A-1.

Figure A-1   The organizational units used in Directory Server

An example of a dn for a user in the AD1 organizational unit would be:

```
uid=l_aduser1,ou=AD1,ou=itso,o=ibm,c=us
```

Schema modification

The objectclass *person* is modified to add an optional attribute `uid`. The Schema was modified using Web Administration Tool as follows:

1. Click **Schema Management** -> **Manage object classes**.

2. Locate the objectclass *person*.

3. Click **Edit** -> **Attributes**.

4. Select **uid** from the drop down list.

5. Click **Add to optional**.

6. Click **OK**.

> **Note:** The same procedure was followed to add the optional attribute givenName to the objectclass *person*.

IBM Lotus Domino Server

Operating System

We install Domino 6.5 (Domino Enterprise Server) on a separate Microsoft Windows 2000 server (with Service Pack 4).

IBM Lotus Domino 6.5 setup

During the install of Domino 6.5, we select the Domino Enterprise Server. In the setup program, we create an ibm.com domain. The organizational unit ITSO is added to this domain. We then add the server, Domino, to this organizational unit. The full dn of the server becomes, **Domino/ITSO/ibm.com**.

All the users are created in the ITSO organizational unit, except for the administrator (Domino Administrator/ibm.com), who is created directly in the ibm.com domain.

Microsoft Active Directory

Operating System

We install Active Directory on a separate Microsoft Windows 2000 server (with Service Pack 4)

Microsoft Active Directory setup

During installation of Active Directory, we create an `itso.ibm.com` domain. We use the Active Directory Users and Computers function to create two organizational units named `IDS1` and `AD1` under domain `itso.ibm.com`. The same tool is used to add any users under the organizational units created.

B

LDAP standards

In this appendix, we provide a list of the LDAP V2 and V3 standards from the Internet Engineering Task Force in the form of Request for Comments.

LDAP standards

Several standards in the form of IETF RFCs exist for LDAP. The following is a brief list of RFCs that apply for LDAP V2 and V3:

- ► RFC 1274 The COSINE and Internet X.500 Schema
- ► RFC 1777 Lightweight Directory Access Protocol (V2)
- ► RFC 1778 String Representation of Standard Attribute Syntaxes
- ► RFC 1779 String Representation of Distinguished Names
- ► RFC 1823 LDAP Application Program Interface (V2)
- ► RFC 2052 A DNS RR for Specifying the Location of Services (DNS SRV)
- ► RFC 2219 Use of DNS Aliases for Network Services
- ► RFC 2222 Simple Authentication and Security Layer (SASL)
- ► RFC 2247 Using Domains in LDAP/X.500 Distinguished Names
- ► RFC 2251 Lightweight Directory Access Protocol (V3)
- ► RFC 2252 Lightweight Directory Access Protocol (V3): Attribute Syntax Definitions
- ► RFC 2253 Lightweight Directory Access Protocol (V3): UTF-8 String Representation of Distinguished Names
- ► RFC 2254 The String Representation of LDAP Search Filters
- ► RFC 2255 The LDAP URL Format
- ► RFC 2256 A Summary of the X.500(96) User Schema for use with LDAPv3
- ► RFC 2596 Use of Language code in LDAP
- ► RFC 2696 LDAP Control Extension for Simple Paged Results Manipulation
- ► RFC 2829 Authentication Methods for LDAP
- ► RFC 2849 The LDAP Data Interchange Format (LDIF) - Technical Specification
- ► RFC 2891 LDAP Control Extension for Server Side Sorting of Search Results
- ► The Open Group schema for liPerson and liOrganization (NAC/LIPS)
- ► Oasis Directory Services Markup Language (DSML) 2.0

Abbreviations and acronyms

ACL	Access Control List	**ITSO**	International Technical Support Organization
ADSI	Active Directory Service Interface	**ITU-T**	International Telecommunications Union - Telecommunications Standardization Sector
ANSI	American National Standards Institute		
API	Application Programming Interface	**LDAP**	Lightweight Directory Access Protocol
CCITT	Consultative Committee on International Telephony and Telegraphy	**LDIF**	LDAP Data Interchange Format
CN	Common Name	**MIME**	Multipurpose Internet Mail Extensions
CRM	Customer Relationship Management	**OID**	Object Identifier
DAP	Directory Access Protocol	**OSI**	Open Systems Interconnect
DIT	Directory Information Tree	**PDC**	Primary Domain Controller
DN	Distinguished Name	**RDN**	Relative Distinguished Name
DNS	Domain Name Service	**RFC**	Request for Comments
DSA	Directory Services Agent	**RID**	Relative Identifier
DSE	Directory Services Engine	**RPC**	Remote Procedure Call
DSML	Directory Services Markup Language	**SASL**	Simple Authentication and Security Layer
ERP	Enterprise Resource Planning	**SID**	Security Identifier
FTP	File Transfer Protocol	**SMTP**	Simple Mail Transfer Protocol
GUI	Graphical User Interface	**SOAP**	Simple Object Access Protocol
GUID	Globally Unique Identifier	**SQL**	Structured Query Language
HR	Human Resources	**SRV**	Service Record (in Domain Name Service)
HTTP	Hypertext Transfer Protocol		
IETF	Internet Engineering Task Force	**SSL**	Secure Socket s Layer
IIOP	Internet InterORB Protocol	**TCP/IP**	Transmission Control Protocol/Internet Protocol
IP	Internet Protocol	**UPN**	User Principal Name
ISO	International Standards Organization	**WINS**	Windows Internet Naming Service
ITDS	IBM Tivoli Directory Server	**XML**	Extensible Markup Language

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information about ordering these publications, see "How to get IBM Redbooks" on page 126. Note that some of the documents referenced here may be available in softcopy only.

► *Using LDAP for Directory Integration - A Look at IBM SecureWay Directory, Active Directory and Domino,* SG24-6136-00

► *Understanding LDAP - Design and Implementation*, SG24-4986-00

► *Active Directory Synchronization with Lotus ADSync*, REDP-0605

Other publications

These publications are also relevant as further information sources, and can be found at:

http://publib.boulder.ibm.com/tividd/td/IBMDirectoryIntegrator5.2.html

► *IBM Tivoli Directory Integrator 5.2: Getting Started Guide,* SC32-1382

► *IBM Tivoli Directory Integrator 5.2: User's Guide,* SC32-1378

► *IBM Tivoli Directory Integrator 5.2: Administrator's Guide,* SC32-1379

► *IBM Tivoli Directory Integrator 5.2: Reference Guide,* SC32-1377

Online resources

These Web sites and URLs are also relevant as further information sources:

► IBM Tivoli Directory Integrator:

http://www-306.ibm.com/software/tivoli/products/directory-integrator/

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

http://www.ibm.com/redbooks

Help from IBM

IBM Support and downloads

http://www.ibm.com/support

IBM Global Services

http://www.ibm.com/services

Index

IBM

Redbooks

Using LDAP for Directory Integration

Using LDAP for Directory Integration

Integration guidelines for systems administrators

Examples to help you integrate directories

Includes IBM Tivoli Directory Integrator

Directories are key for a successful IT operation and e-business application deployments in medium and large environments. IBM understands this requirement and supports it by providing directory implementations based on industry standards at no additional cost on all its major platforms and even important non-IBM platforms. The IBM Directory Server implements the Lightweight Directory Access Protocol (LDAP) standard that has emerged quickly in the past years as a result of the demand for such a standard.

We concentrate on some advanced LDAP tasks, such as referrals and schema extensibility. We also provide a scenario-based approach to discuss Directory Integration, using some of the leading directory products available: the IBM Directory Server, IBM Directory Integrator, Lotus Domino, and Microsoft's Active Directory.

This redbook will help you understand, install, and configure the IBM Directory Server into a heterogeneous repository environment. It is targeted at system specialists who need to know the concepts and the detailed instructions for a successful Meta directory implementation.