EVIDEN

Identity and Access Management

# DirX Directory

## Administration Guide

Version 9.1, Edition June 2025

# Table of Contents

# Preface

The *DirX Directory Administration Guide* describes basic DirX Directory administration tasks and how to perform them with the DirX Directory administration tools. The book is structured as follows:

- Chapter 1 provides information about DBAM and storage management.
- Chapter 2 introduces the functions of the DirX Directory administration tools.
- Chapter 3 provides a simple example of how to set up the DirX Directory service and populate the database with directory information.
- Chapter 4 describes the fundamental administrative tasks involved in setting up the DirX Directory service and populating the directory information tree (DIT).
- Chapter 5 describes extensions that can be made to the DirX Directory service.
- Chapter 6 describes how to set up a shadow DirX Directory service.
- Chapter 7 describes how to distribute a DIT between DSAs.
- Chapter 8 describes how to set up a directory service based only on shadowing in which pieces of a DIT held by two different DSAs are replicated to each DSA.
- Chapter 9 describes how to set up a mix of synchronous and asynchronous shadow DirX Directory services.
- Chapter 10 describes how to set up and manage a floating master-shadow configuration that distributes DAP processing requests from LDAP servers across the shadow DSAs in the configuration to provide for better DAP load balancing.
- Chapter 11 describes how to configure the DSA to export and import directory data to and from LDIF files.
- Chapter 12 describes the DirX Directory administration tools that enable you to monitor a DSA's performance, set up audit trails, and perform program tracebacks.
- Chapter 13 describes how to use external authentication mechanisms.
- Chapter 14 provides additional information about miscellaneous issues.

# DirX Directory Documentation Set

DirX Directory provides a powerful set of documentation that helps you configure your directory server and its applications.

The DirX Directory document set consists of the following manuals:

- *DirX Directory Introduction*. Use this book to obtain a description of the concepts of DirX Directory.
- *DirX Directory Administration Guide*. Use this book to understand the basic DirX Directory administration tasks and how to perform them with the DirX Directory administration tools.
- *DirX Directory Administration Reference*. Use this book to obtain reference information about DirX Directory administration tools and their command syntax, configuration files, environment variables and file locations of the DirX Directory installation.
- *DirX Directory Syntaxes and Attributes*. Use this book to obtain reference information about DirX Directory syntaxes and attributes.
- *DirX Directory LDAP Extended Operations*. Use this book to obtain reference information about DirX Directory LDAP Extended Operations.
- *DirX Directory External Authentication*. Use this book to obtain reference information about external authentication.
- *DirX Directory Supervisor*. Use this book to obtain reference information about the DirX Directory supervisor.
- *DirX Directory Plugins for Nagios*. Use this book to obtain reference information about DirX Directory plugins for Nagios.
- *DirX Directory Disc Dimensioning Guide*. Use this book to understand how to calculate and organize necessary disc space for initial database configuration and enhancing existing configurations.
- *DirX Directory Guide for CSP Administrators*. Use this book to obtain information about installing, configuring and managing DirX Directory in the context of a Certificate Provisioning Service operating in accordance with regulations like the German “Signaturgesetz”.
- *DirX Directory Release Notes*. Use this book to install DirX Directory and to understand the features and limitations of the current release.

# Notation Conventions

**Boldface type**
In command syntax, bold words and characters represent commands or keywords that must be entered exactly as shown.

In examples, bold words and characters represent user input.

*Italic type*
In command syntax, italic words and characters represent placeholders for information that you must supply.

[ ]
In command syntax, square braces enclose optional items.

{ }
In command syntax, braces enclose a list from which you must choose one item.

In Tcl syntax, you must actually type in the braces, which will appear in boldface type.

|
In command syntax, the vertical bar separates items in a list of choices.

...
In command syntax, ellipses indicate that the previous item can be repeated.

*install_path*
The exact name of the root of the directory where DirX Identity programs and files are installed. The default installation directory is *userID_home_directory**/DirX Identity* on UNIX systems and **C:\Program Files\DirX\Identity** on Windows systems. During installation the installation directory can be specified. In this manual, the installation-specific portion of pathnames is represented by the notation *install_path*.

# 1. Understanding DBAM and Storage Management

DBAM requires that physical disk device storage be allocated to it to store the directory database and the transaction logs. The following tasks need to be performed by you and/or other administrators at your site before you can set up the DBAM database:

1. First, you need to examine your directory database requirements and determine how much disk storage you need for the DBAM database.
2. Then, you must decide on a fault-tolerant strategy for maximizing the DBAM database's availability and reliability.
3. The next step is to allocate storage for DBAM on the operating system platform on which DirX Directory is to run according to your database requirements.
4. Finally, you need to describe the allocated disk storage to the DBAM database.

This chapter provides background information about disk technologies that may help you to understand the implications and requirements of the DBAM database on the physical disk configuration at your site. It briefly describes:

- The anatomy of a disk device. This information provides an introduction to the components and terminology used to describe a disk device.
- Redundant Array of Disks (RAID) structure and management. This information will help you to understand RAID-based fault-tolerant configurations that can be used with DirX Directory to maximize its availability and reliability. It also makes recommendations about the fault-tolerant configurations best suited to DirX Directory.
- The disk management architectures and tools of the two operating systems on which DirX Directory can be installed: Microsoft Windows and Linux. This information will help you to understand how the DirX Directory host operating system's physical disks can be configured and the process for configuring them.

The chapter also describes the DBAM "view" of the underlying disk storage and introduces the tools you use to configure it to the storage you have allocated for its use.

## 1.1. Disk Hardware Concepts

A magnetic disk device consists of a set of **platters** on which data resides and a set of **disk arms** with **heads** that read and write data to and from the platters.Data is stored on both sides of a platter.Each track arm-head pair reads/writes one side of the platter.For example, head 0 controls the top side of platter 0 (the topmost platter), and head 1 controls the bottom side of platter 0. The following figure illustrates these components.

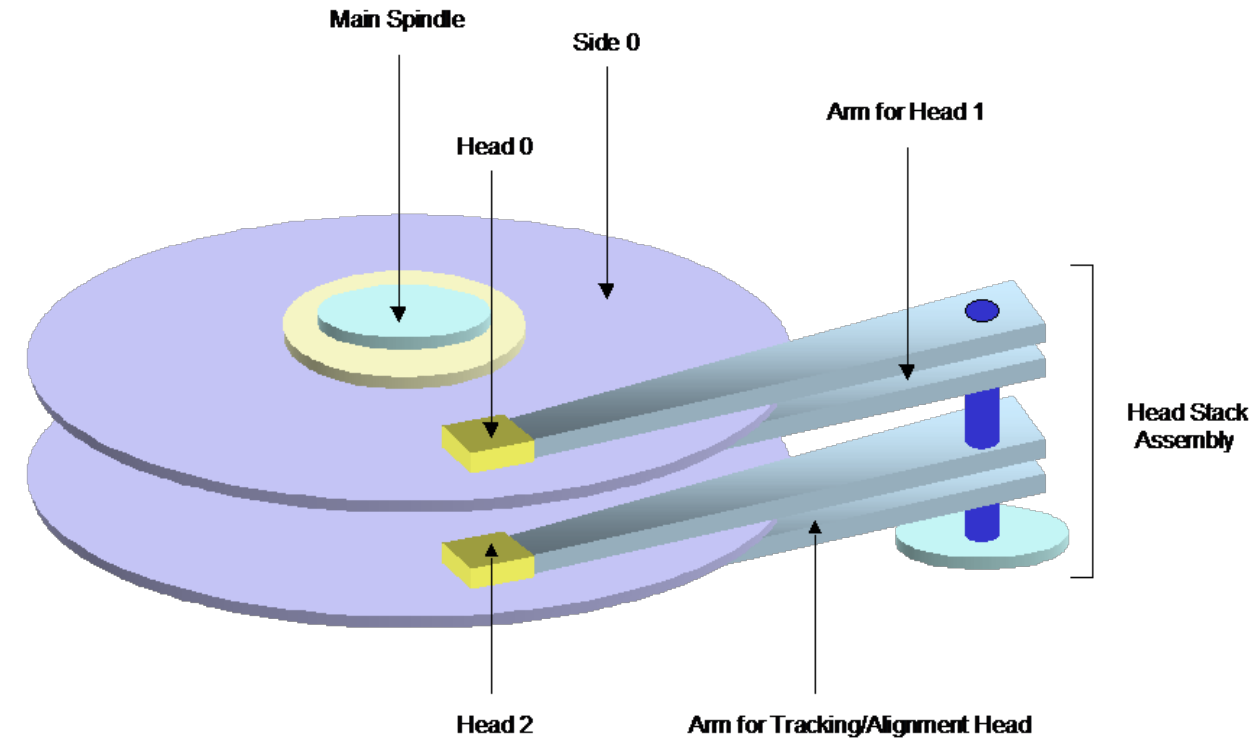



*Figure 1. Platter, Disk Arm, and Head Relationships*

A platter consists of a number of **tracks**, for example 1024. Tracks exist on both sides of the platter.A **cylinder** represents the top and bottom tracks on each platter.For example, cylinder 0 consists of the outermost top and the bottom track on the platter.The following figure illustrates these components.

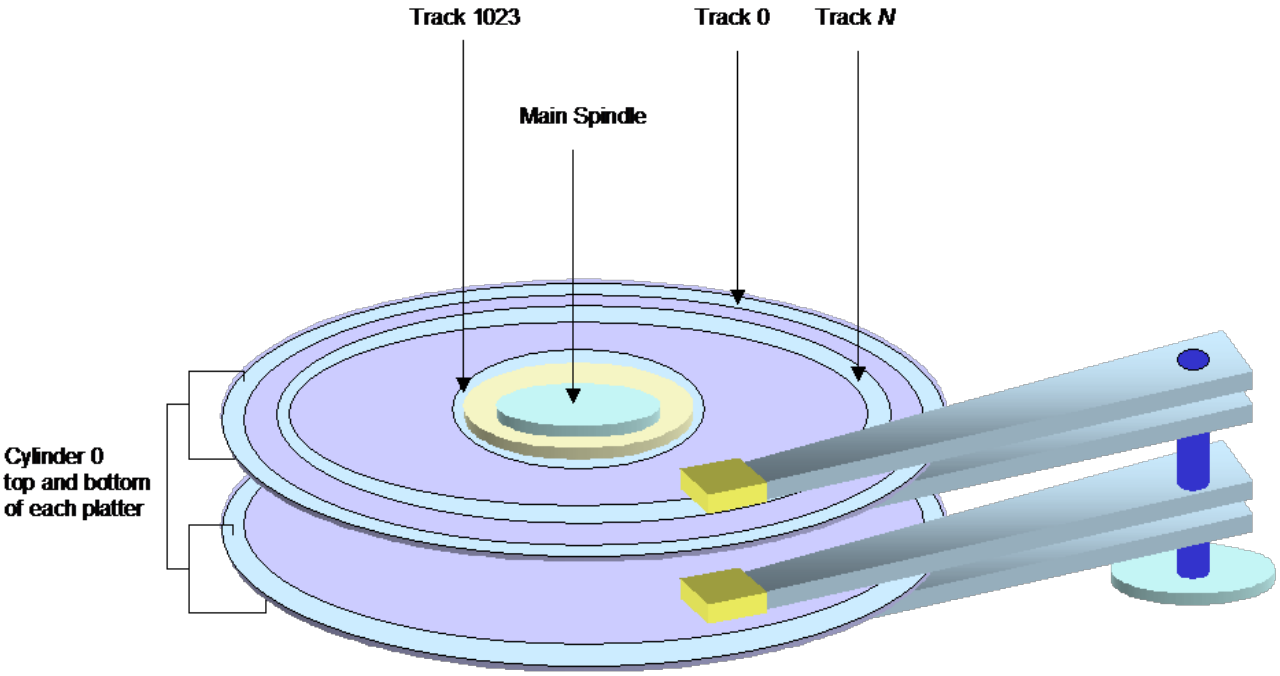
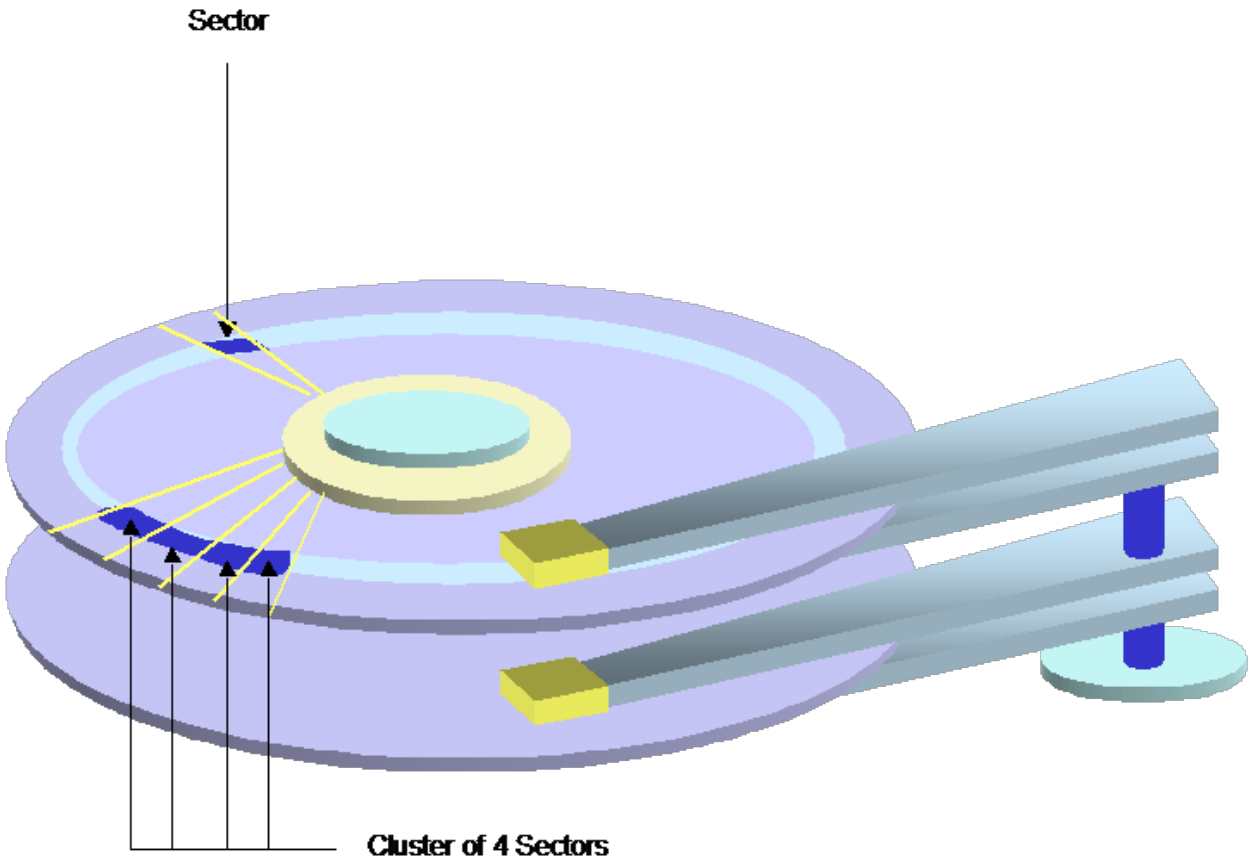



*Figure 2. Platter, Track and Cylinder Relationships*

A track is divided into **sectors**.A sector is the smallest physical storage unit on a disk.It is almost always 512 bytes in size.Operating systems use sectors in **clusters**.The following figure illustrates these components.

*Track, Sector and Cluster Relationships*




The operating system managing the disk device translates the cylinder, head, and sector physical format into a logical format.

## 1.2. RAID Disk Management

A RAID system consists of an intelligent manager, either implemented in hardware or software, that can manage multiple disk drives so that the system can withstand the failure of any individual member without a loss of data.RAID provides a method of accessing multiple individual disks as if the array of disks is one large disk.Data access is spread over these multiple disks, reducing the risk of losing all data if one drive fails, and improving access time.RAID improves availability, but it cannot replace backup.

### 1.2.1. RAID Levels

The RAID Advisory Board has created industry standards for levels of RAID functionality based on the 1988 University of California, Berkeley RAID papers.Basic RAID levels of interest to this discussion include:

- Disk striping (RAID level 0)
- Disk mirroring (RAID level 1)
- Disk striping and mirroring (RAID level 0+1, or 10)
- Distributed parity RAID (RAID level 5)

#### 1.2.1.1. Disk Striping (RAID-0)

Disk striping is a performance-oriented data mapping technique that provides no fault tolerance at all. Data is written in blocks across multiple disks so that one drive can be writing or reading a block while the next drive is seeking the next block. The following figure illustrates disk striping.

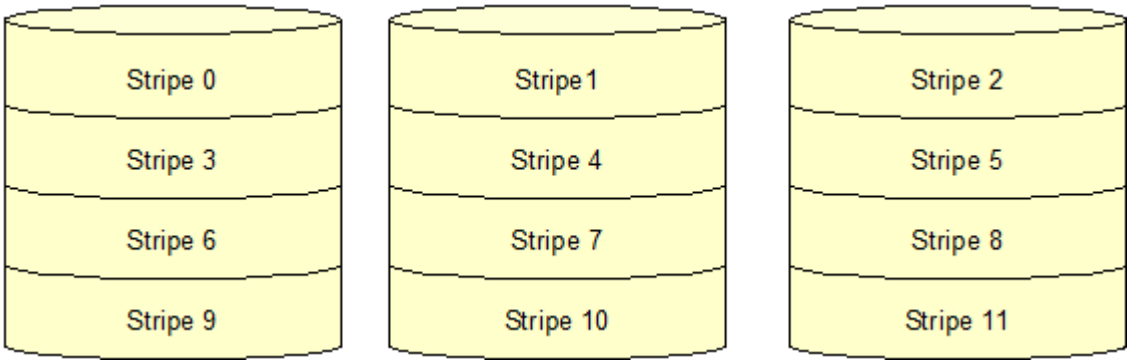


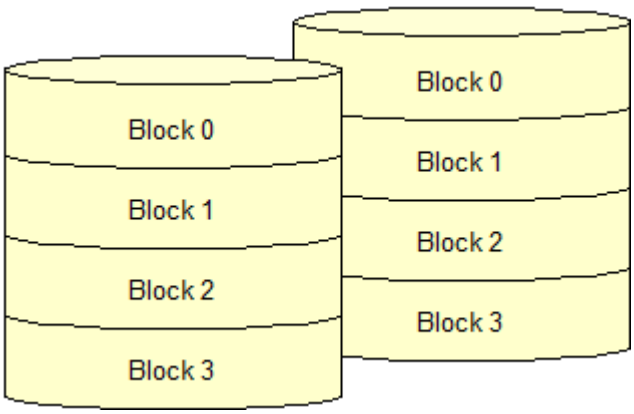

*Figure 3. Disk Striping (RAID 0)*

Disk striping provides a higher access rate and full use of the disk array's capacity. However, if one disk fails, the entire group fails and data cannot be accessed until the disk has been repaired and the data has been restored.

#### 1.2.1.2. Disk Mirroring (RAID-1)

Disk mirroring provides an identical twin (or more) for a selected disk. Data is written twice—once to each disk. If there is a read failure on one of the disks, the RAID system can read the data from the other disk in the mirror set. The following figure illustrates disk mirroring.

*Disk Mirroring (RAID 1)*




Disk mirroring provides very good fault tolerance. It may give better read performance because data can be read from both (or all) disks in parallel, but writes must go to several disks, so writes can be more expensive. However, only half the available disk space can be used for storage, as the other half is needed to create the mirror. This makes the mirrored disk configuration more expensive to implement.

#### 1.2.1.3. Mirrored Striped Set (RAID 0+1)

A mirrored striped set combines both disk mirroring and disk striping without parity. When data is written to a mirrored striped set, two mirrored virtual disks are created instead of just one virtual disk, as with striping. This configuration provides fast data access, like disk striping and single-disk fault tolerance, like disk mirroring. However, like mirrored disks, only half the available disk space can be used for storage; the other half must be used to create the mirror.

#### 1.2.1.4. Distributed Parity (RAID-5)

Parity adds fault tolerance to disk striping by including parity information with the data for error recovery. In Distributed Parity (RAID level 5), the parity information is distributed on the different disks (rather than being contained on a dedicated disk, as is the case with RAID levels 3 and 4). The parity information is used to recover data if one disk fails.

The disadvantage of this configuration is a slower write cycle (*n*-2 reads and 2 writes for each block written). The disk array capacity is *n*-1, and requires a minimum of 3 disks. Whereas RAID 0+1 requires 100% capacity to protect the data, RAID-5 requires as little as 50% (2+1) and commonly only 20% (5+1).

### 1.2.2. Recommendations for a DirX Directory Fault Tolerant Configuration

As mentioned earlier, RAID systems can be implemented as software-based systems or hardware-based systems.If you plan to provide a fault-tolerant environment for DirX Directory operation (it is an option, not a requirement), we recommend the following:

- Choose a hardware-based fault-tolerant solution.There is a wide choice of hardware-based fault-tolerant solutions offered by both computer manufacturers and third party suppliers.
- Implement the RAID 0+1 configuration.However, if disk economy is an issue, implement the RAID-5 configuration (although disks are inexpensive).
- Build in hardware redundancy for all components: controllers, buses, and so on.

You **must** provide an uninterruptible power supply (UPS) for the RAID configuration.Otherwise, the RAID system's disk cache may lose data.There is a large performance penalty on Windows without disk caching.

## 1.3. Windows Disk Management

A physical disk managed by Windows is either a **basic disk** or a **dynamic disk**.When you first install Windows, whether it is a new installation or an upgrade from Windows NT, all disks on the system are configured as basic disks.You can convert a basic disk to a dynamic disk with the Windows Disk Management snap-in tool.

### 1.3.1. Basic Disks

A basic disk is composed of one or more **partitions**.A basic disk can contain a maximum of

4 partitions.A partition is composed of one or more **volumes** (also called logical drives).

A partition can be a **primary partition** or an **extended partition**.A primary partition contains a single volume that is the size of the partition.An extended partition can contain more than one volume.The combined size of all the volumes on an extended partition must be less than or equal to the size of the partition.

Basic disks are backward compatible with earlier versions of Windows (for example, Windows NT 4.0 and Windows 98) and with MS/DOS.

### 1.3.2. Dynamic Disks

Dynamic disks are not available on notebook computers, and they are not supported on removable disks or on disks using Universal Serial Bus (USB) or FireWire (IEEE 1394) interfaces.

A dynamic disk contains **volumes**; there is no concept of partitions or logical devices. A dynamic disk has the following advantages:

- Unlike a basic disk, which is limited to four partitions, a dynamic disk can contain an unlimited number of volumes. You can create as many volumes on a dynamic disk as there is disk space available.
- You can extend volumes by adding free space to them.
- On multiple disk systems, you can combine space from more than one disk into special volume configurations.
- If you want to use the fault-tolerant features provided by the operating system as software RAID support, such as mirroring (RAID-1) or striping with parity (RAID-5), you must use dynamic disks.

As with basic disk partitions, you can assign drive letters or drive paths to volumes when you create them (however, drive paths must be used for disks in a DBAM configuration.)

A volume can be:

- A simple volume, which is a single disk.
- A spanned volume, which contains up to 32 separate disks. Data fills the space on the first disk, and then the next, and so on.
- A mirrored volume, which contains two disks that each contain an identical copy of the volume to provide for fault tolerance.
- A striped volume, which contains up to 32 separate disks. Data is spread evenly between all of the disks to provide disk-access efficiency.
- A RAID-5 volume, which consists of three or more separate disks. Data and parity information are spread evenly over the disks to provide for fault tolerance.

Simple and spanned volumes can be extended: you can add available free space to increase their size.

RAID-5 volumes can only be created on Windows server computers.

Volumes and basic disk partitions cannot co-exist on the same physical disk, but you can have both dynamic and basic disks on the same system.

When creating volumes, keep in mind that:

- A volume is a logical area that resides on one (simple volume) to several (spanned or striped volume) disks.
- A disk can be divided into one or several volumes.

The following figures illustrate these concepts and how they relate to the DBAM data storage model (which is described in detail later on in this chapter in the section "DBAM Storage Model"). The first figure shows one disk with two simple volumes, one for directory data and one for transaction data.

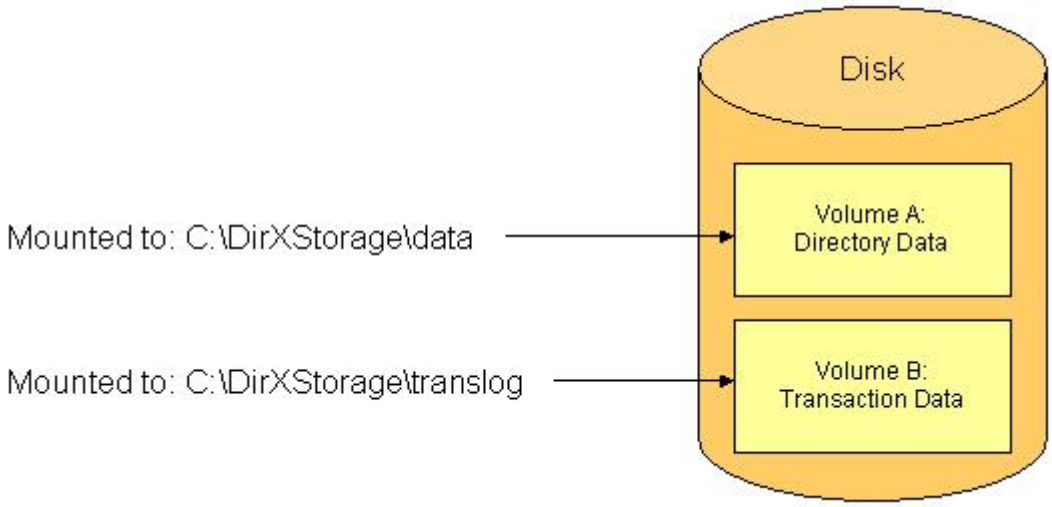



*Figure 4. One Disk, Two Simple Volumes*

The second figure illustrates three disks with one spanned or striped volume for directory data and one simple volume for transaction data.

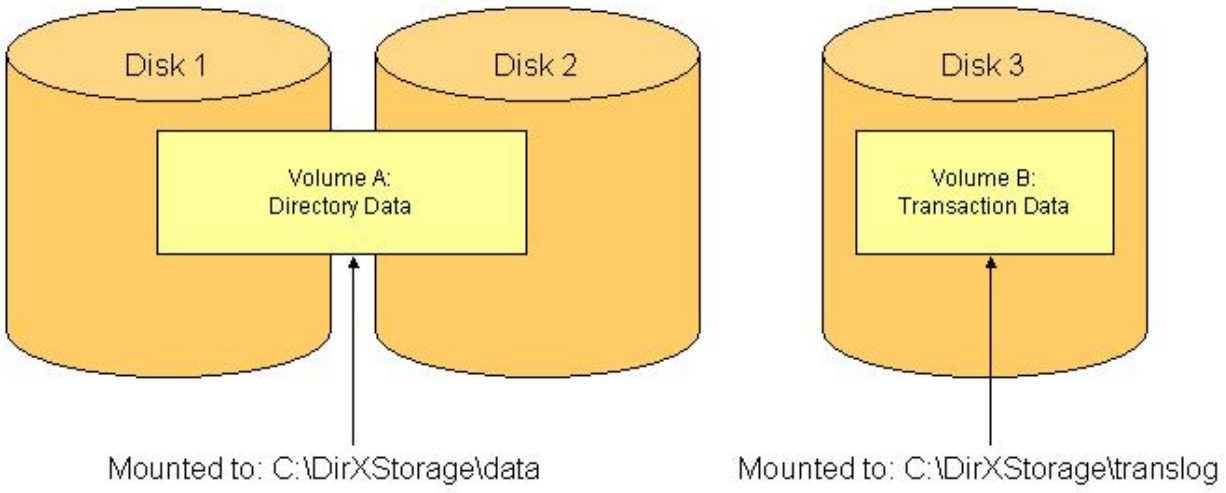



*Figure 5. Three Disks, One Simple Volume, One Spanned Volume*

### 1.3.3. Windows Disk Management Tool

Administrators use the Disk Management snap-in for the Microsoft Management Console (MMC) to manage basic and dynamic disks on Windows. To access the Disk Management snap-in:

1. Right-click the **Start** button, and then click **Computer Management**.
2. Expand the **Storage** folder in the hierarchical tree view and click the **Disk Management** folder. The Disk Management snap-in displays information about the disk devices available to the operating system, as shown in the following figure.

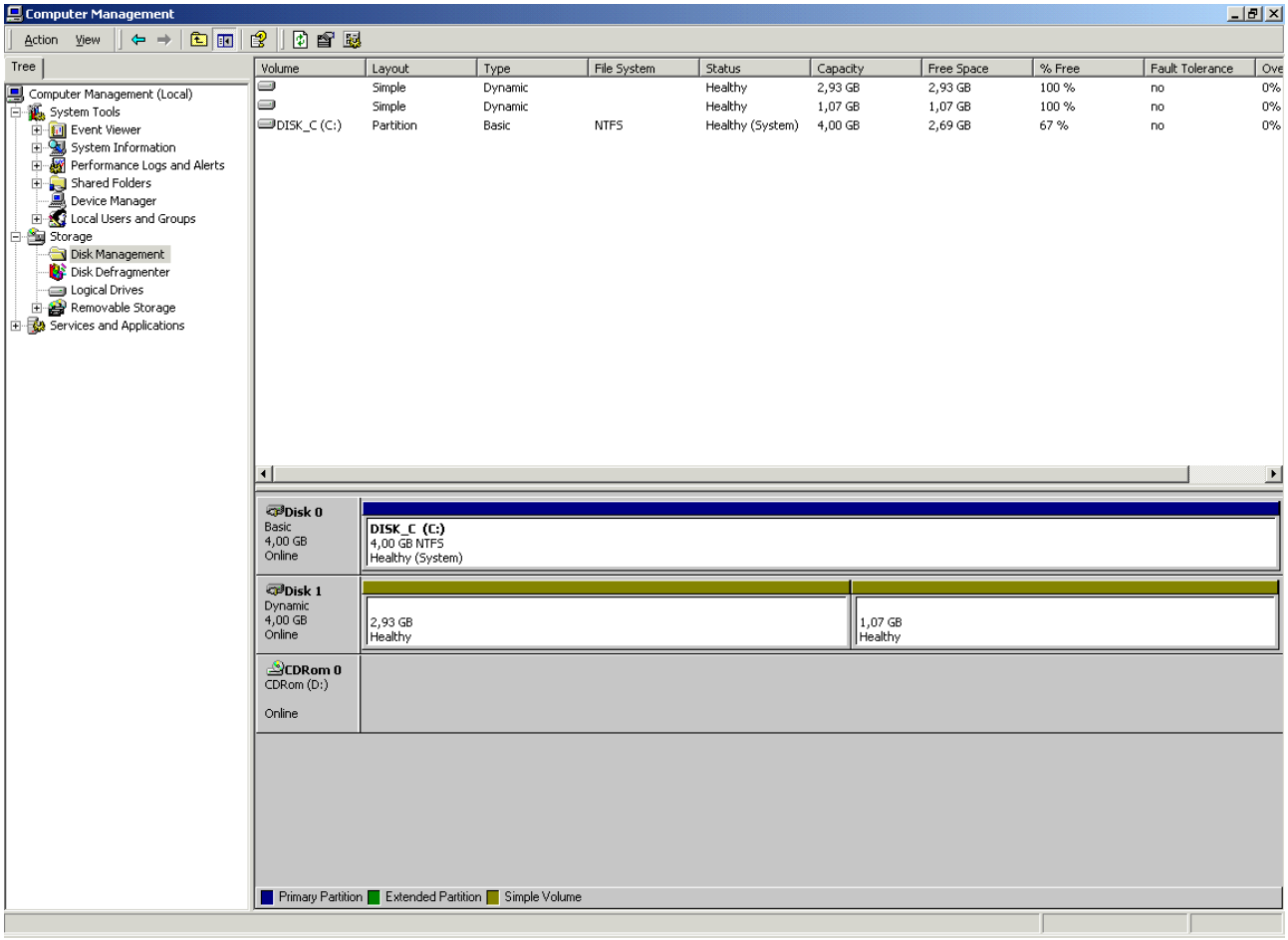



*Figure 6. Disk Management Window*

When you make changes to the disk configuration with Disk Management, you do not need to restart your computer for your changes to take effect. The MMC online help provides more information about how to use MMC and the Disk Management snap-in.

### 1.3.4. DBAM Requirements for Windows Disks

Windows disks to be used for DBAM data storage have the following requirements:

- It is recommended to use dynamic disks (either upgraded from basic disks or created from scratch)
- In order to get best performance the disk "write cache" feature must be enabled. This implies the usage of an uninterruptable power supply (USP). The "write cache" can be

switched on under "Computer Management", "Device Manager", "Disk Drives" and then in each concerned device in the tab "Disk Properties".

- Volumes created for DBAM on these disks:
  - Must be assigned a drive path (mounted at an empty folder in an NTFS partition).This action gives the volume a name that can be used in DBAM configuration.(DBAM does not support drive letters).
  - Must not be formatted

In addition, the machine on which the DBAM database is to reside must run under the control of an uninterruptible power supply (USP) or within a computer center that guarantees an uninterrupted power supply.

Note that RAID devices appear to the Windows disk management system as normal disks and are detected automatically when the system is booted.

## 1.4. Linux Disk Management

Refer to your system documentation to get information about Linux disk management.

## 1.5. The DBAM Storage Model

In the DBAM model, the raw devices—the volumes on Windows dynamic disks or the raw data slices (slices that do not contain a file system) on Linux disks—that comprise its database storage are one of two types:

- A **transaction device**, which stores transaction logs.The DBAM database requires that you configure one (and only one) volume or slice as a transaction device.
- A **directory data device**, which stores directory data.The DBAM database requires that you configure at least one unformatted volume or non file-system slice as a directory data device.However, you can configure up to six volumes or slices to split directory data over several disks, according to their respective purposes, to improve performance.

You use the DBAM configuration and initialization tools to configure unformatted volumes or non file-system slices as DBAM devices.

A DBAM data device can store the following kinds of directory data:

- Real directory object data—the complete information about the object, including its relative distinguished name (RDN), a pointer to its parent object, its attributes and their values.
- Pseudo object data—references to the "real" objects used internally by the DBAM database.
- Attribute value index data—the references to the "real" objects that contain the corresponding attribute value.
- Bit string data—the bit string references for bitmapped indexes.
- Tree data—the hierarchical relationships between the objects.

- General data—storage administration data that does not have a specific purpose.

You allocate directory data to a data device according to its type. You can allocate all data types on a single DBAM data device, or you can use multiple data devices to hold the different types of data. For example, you can allocate real object data to one data device, allocate attribute index data to a second data device, and allocate the remaining types of data to a third data device. Each data device maps to a raw device. You can create a maximum of six data devices on six different raw devices, with each data device allocated to one DBAM data type. However, you cannot allocate the same DBAM data type on two different data devices (raw devices).

You can also allocate DBAM data types to a data device according to size; for example, you can specify that 5GB of a data device is to be allocated to tree data, and the remaining disk space is to be divided up between pseudo object, bit string, and general data.

Thus, when you configure a raw device as a data device, you configure it according to the type(s) of directory data it will store and the amount of storage to be allocated to the type(s).

The following figure illustrates the DBAM storage hierarchy.

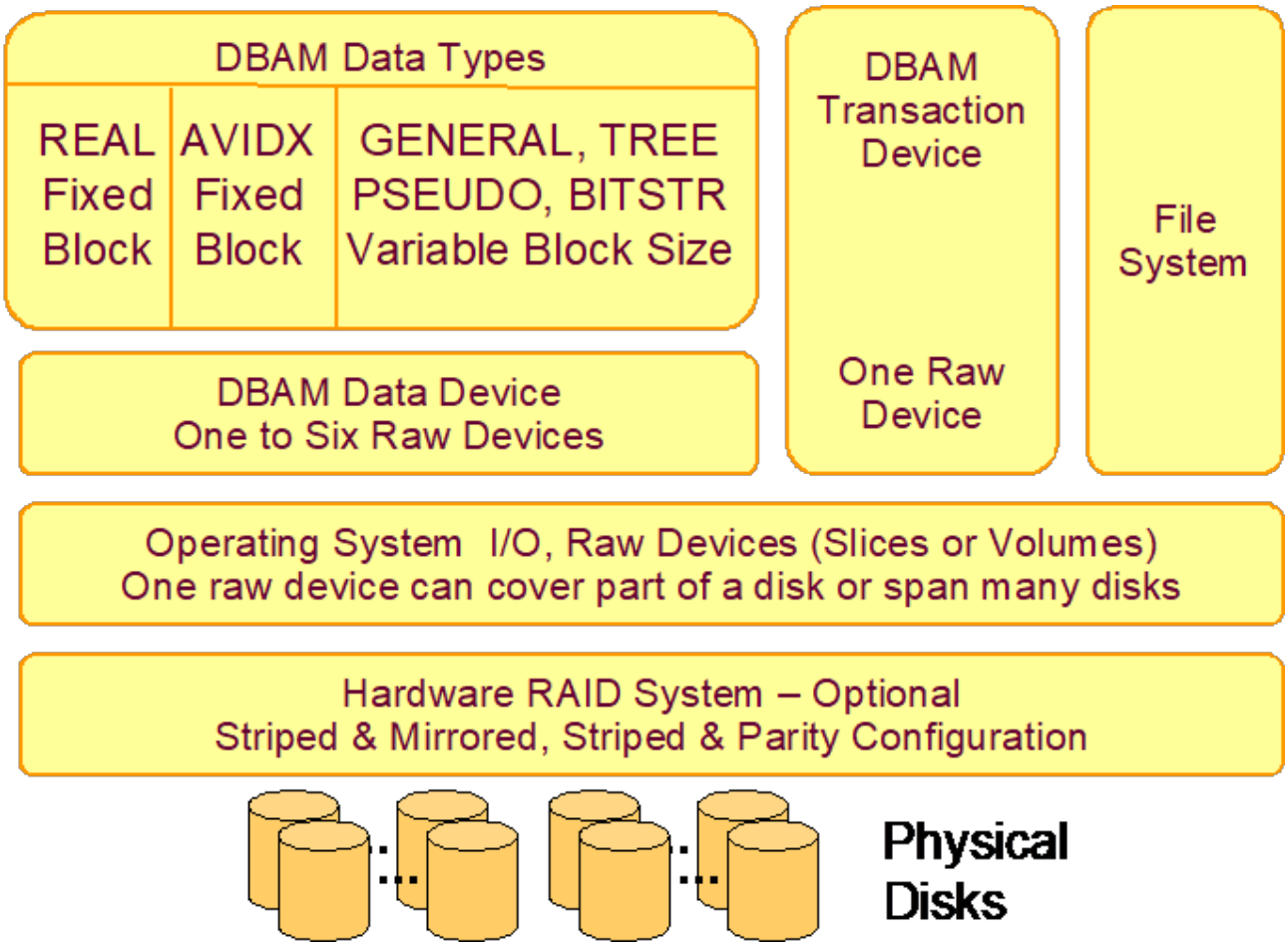



*Figure 7. DBAM Storage Hierarchy*

In the figure:

- The DBAM data types are allocated to one to six DBAM data devices, which map to one

to six raw devices.

- The DBAM transaction device maps to one raw device.
- The file system is separate from the DBAM devices.It contains the DirX Directory installation, and stores "normal" files such as journal, audit, work area and backup files.
- The raw devices are defined at the operating system level using the operating system tools.
- The optional RAID system is connected to the operating system to manage striped and mirrored, striped and parity configurations.

As noted in the figure, the block size of each real and attribute value index data type is fixed, whereas the block size of the other data types is variable.The following figure illustrates a sample DBAM disk configuration to manage 30 million directory entries in a mirrored disk configuration.

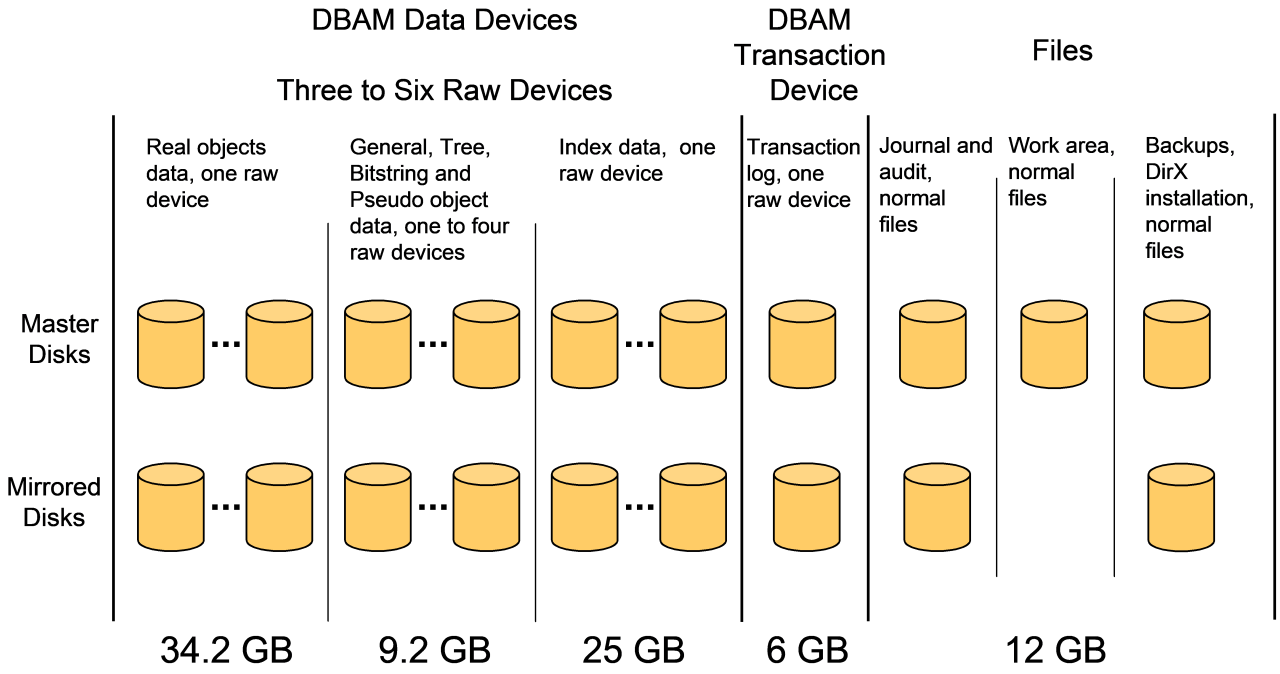



*Figure 8. Example DBAM Disk Configuration*

# 1.6. DBAM Configuration Tools

To configure the DBAM database, you first use the DBAM configuration tool **dbamconfig** to create a configuration of raw devices to be used as the data device(s) and transaction device and save it as a "database profile".You then supply this profile to the DBAM initialization tool **dbamboot**, which uses the information in the profile to initialize the DBAM database and set up the raw device mapping to the DBAM data and transaction devices.Alternatively, you can use the **dbaminit** command allocate physical disk space in the file system for the data and transaction devices instead of using raw devices.

Instead of performing a **dbamconfig** and a **dbamboot** (or **dbaminit**) command you can perform a **dirxconfig** command that first performs a **dbamconfig** and then a **dbamboot** (or **dbaminit**) command with pre-defined values.

### 1.6.1. dirxconfig

The **dirxconfig** tool is a command-line tool (accessible from the MS/DOS prompt in Windows) that is co-located with the DBAM database and the DSA.The **dirxconfig** tool internally performs first a **dbamconfig** and then a **dbamboot** command with pre-defined values.It distinguishes between values for a small, a mid-size and a large configuration.

### 1.6.2. dbamconfig

The **dbamconfig** tool is a command-line tool (accessible from the MS/DOS prompt in Windows) that is co-located with the DBAM database and the DSA. You use **dbamconfig** to create, delete and display DBAM **database profiles**, which are structures that link the allocated raw devices to the DBAM data device/transaction device format. These profiles are configuration information and are not part of the database data. They are stored in the registry on Windows and in a file on Linux.

When you use **dbamconfig** to create a profile, you specify one raw device as a transaction device and up to six raw devices as data devices. If you specify a single data device and do not make specific directory type allocations, **dbamconfig** uses the following default allocation:

- Real object data is allocated to 40% of the device
- Attribute value indexes are allocated to 40% of the device
- The other data types are allocated to the remaining 20% of the device

The *DirX Administration Reference* provides command syntax details and examples.

### 1.6.3. dbamboot

The **dbamboot** tool is a command-line tool (accessible from the MS/DOS prompt in Windows) that is co-located with the DBAM database and the DSA. You use the **dbamboot** tool to initialize the DBAM database according to a DBAM profile created with **dbamconfig**. You also use the **dbamboot** tool to define the size of individual real object data blocks and the number of indexed attribute types. You can also use the **dbamboot** tool whenever you want to re-initialize (delete) your database. The *DirX Administration Reference* provides command syntax details and examples.

### 1.6.4. dbaminit

The **dbaminit** tool is a command-line tool (accessible from the MS/DOS prompt in Windows) that is co-located with the DBAM database and the DSA. You use the **dbaminit** tool to initialize a file-based DBAM database; that is, you allocate physical disk space in the file system for the DBAM data device(s) and transaction device. When you use **dbaminit** to configure the database, you specify the name and size of a file to be used as the transaction device and name(s) and size(s) of file(s) to be used as the data device(s). The *DirX Administration Reference* provides command syntax details and examples.

# 2. Using the DirX Directory Administration Tools

This chapter summarizes the features and functions of the DirX Directory administration tools and explains when and why a DirX Directory administrator will want to use each one.

On Windows Server, you need to run the DirX Directory administration tools with administrator rights. Use **Run as...** from the context-sensitive menu to run the command prompt as Administrator.

## 2.1. Using DirX Directory Manager

DirX Directory Manager is a DirX Directory administration tool that you can use on Windows or Linux systems to access the DirX directory service through a DirX Directory LDAP server that is running on a local or remote Windows or Linux system.DirX Directory Manager is an LDAP client written in the Java programming language.It is capable of accessing any directory service that runs an LDAP server, not just DirX Directory.

For DirX Directory, you can use DirX Directory Manager to:

- Manage the directory entries in the service's Directory Information Tree (DIT)
- Manage the directory schema
- Manage shadowing and LDIF agreements
- View and edit the Attribute Indices
- View and edit the collective attribute subentries
- View and edit the LDAP configuration subentries
- View and edit the password policy subentry
- View and edit the proxied authorization control subentry
- View and edit the access control subentries
- Perform SSL authentication
- Create, edit, and run Tcl scripts in a convenient way
- View audit log files
- View LDAP and DSA Monitoring Information

The sections that follow briefly describe DirX Directory Manager's graphical user interface and its functions. The DirX Directory Manager online help provides detailed usage information. The DirX Directory Manager release notes for DirX Directory provides information on how to start DirX Directory Manager on Windows and Linux systems.

### 2.1.1. When to Use DirX Directory Manager

Although DirX Directory Manager is intended for use mainly as a directory administration tool, both directory users and DirX Directory administrators can use it to browse and search

the DirX Directory database. For DirX Directory administrators, DirX Directory Manager can be used as a dialog-based alternative to the **dirxmodify** command-line tool for performing small-scale schema administration. DirX Directory Manager also allows DirX Directory administrators to model a schema from an easy-to-use, dialog-based interface, and then export the schema to an LDIF file for future loading to an LDAP server via DirX Directory Manager, the **dirxload** or **dirxmodify** command line tools or an installation procedure.

### 2.1.2. Using the DirX Directory Manager Welcome View

When you start DirX Directory Manager, it displays its Welcome view. This view displays a list of LDAP servers to which you can connect. DirX Directory LDAP servers are identified by the text "DirX" in the Server Type field. To connect to an LDAP server, double-click it in the list, or select it and then click **Open**. If the selected LDAP server requires authenticated connection, DirX Directory Manager next displays a login dialog. Enter your distinguished name (DN) and password in the fields provided. When you have successfully connected to the LDAP server (anonymously or authenticated), DirX Directory Manager's main window appears.

You can also use the Welcome view to:

- Manage LDAP server profiles; the section "Adding LDAP Servers to DirX Directory Manager" describes this task in more detail
- View the documentation that is available with DirX Directory Manager
- Manage scripts with the script manager (the online help describes this task in more detail)

### 2.1.3. Using the DirX Directory Manager Main Window

DirX Directory Manager's main window represents LDAP servers as "view groups". The main window's view bar displays the DirX directory service view group and the view groups of any other LDAP directories that you (or another administrator) have added to DirX Directory Manager. You can click a view group to display its views or use the DirX Directory Manager menu bar to display them.

From the DirX Directory view group, you can select from four different views:

- The Directory Entries view, which allows you to view and edit user directory entries and their attributes and perform simple and complex searches on entries and attributes
- The Quick Search view, which allows you to search for user directory entries using selected search filters and display the results of the search (entry name and a subset of its attributes)
- The Configuration view, which allows you to view and edit administrative directory entries and their attributes, such as LDAP server configuration subentries, access control, collective attribute subentries, proxied authorization control subentries, the password policy subentry, and context prefixes
- The Schema view, which allows you to view and edit the object classes and attribute types defined in the DirX Directory schema using a hierarchical tree view, and manage attribute indexes (Database)

- The Replication view, which allows you to view and edit all shadowing and LDIF agreements and to perform the switch operation
- The Monitoring view, which allows you to display statistics and network management information (NMI) data collected by the DirX Directory DSA and LDAP server.

The following figure shows the DirX Directory view group with the Directory Entries view open.

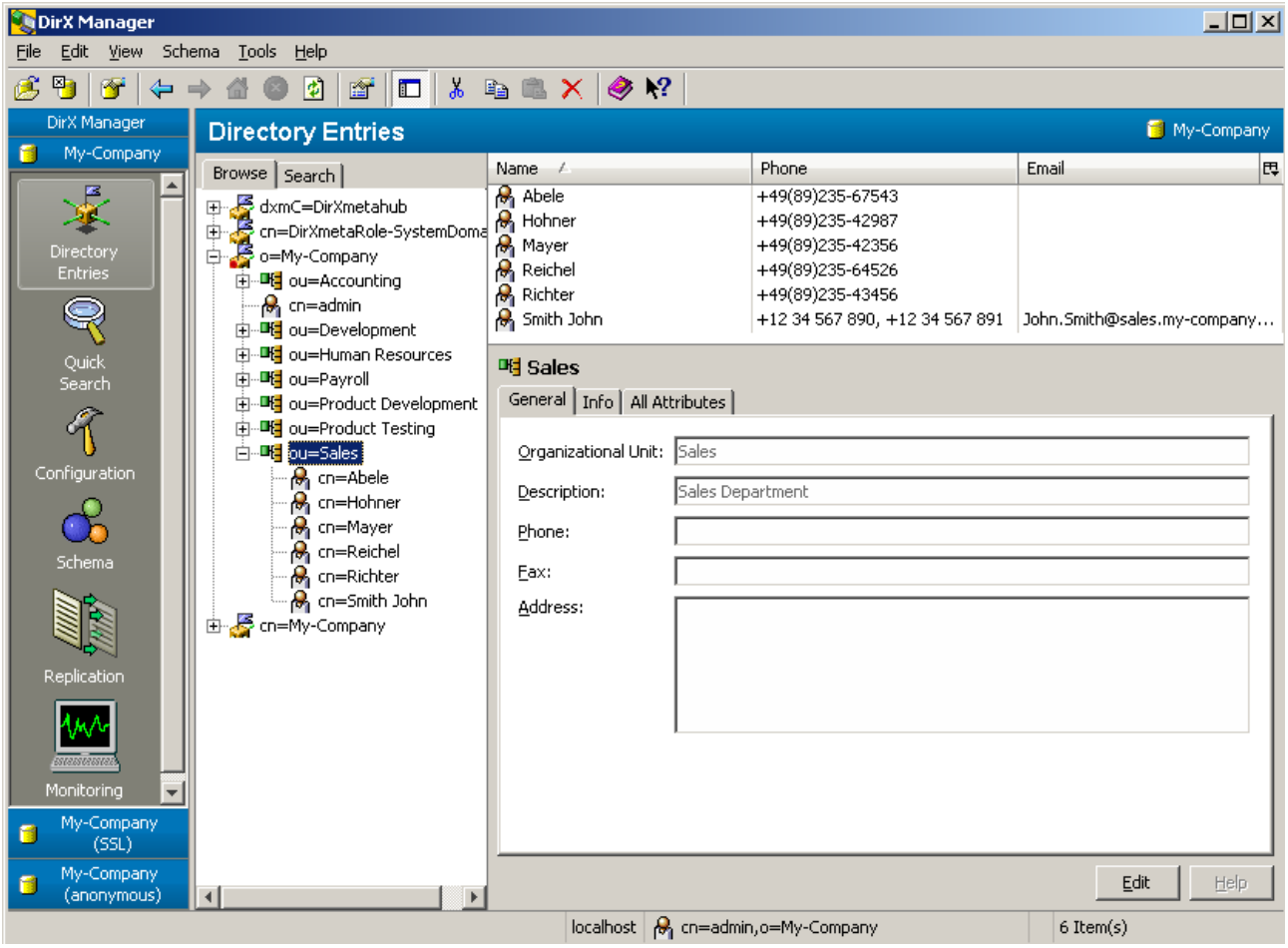



*Figure 9. DirX Directory Manager View Group*

By default, authenticated users (users who have connected to the DirX Directory LDAP server with a distinguished name and password) are permitted to use any of these views, while anonymous users can only use the directory entries and quick search views. DirX Directory administrators can use DirX Directory Manager to change this default.

#### 2.1.3.1. Using the Directory Entries View

The DirX Directory Entries view consists of a browse pane and a search pane. The browse pane displays a hierarchical tree view of the directory entries in the DIT on the left, and the attributes of a selected entry in the tree on the right. This view of the DIT displays only those entries that are typically visible to users, such as directory users and groups; administrative entries are not displayed. From the Browse pane, you can:

- Browse the directory entries in the DIT

- Create, copy, rename and delete individual directory entries and subtrees of entries
- Modify the attributes of directory entries
- Move individual directory entries and subtrees of entries to different locations in the DIT

The search pane consists of two windows: a window with fields that permit you to perform both simple and complex searches on entries and attributes in the DIT, and a window that displays the search results.

You can configure DirX Directory Manager's search function to:

- Set limits on the amount of time a search can take and the number of entries that can be returned
- Ignore or automatically follow referrals returned by the LDAP server
- De-reference all aliases (entries that refer to other entries), de-reference only those aliases encountered in search bases, de-reference only those aliases returned in a search result, or perform no de-referencing at all

You make these settings on a per-server basis via a DirX Directory Manager server profile, which permits you to maintain different settings for each LDAP server with which DirX Directory Manager communicates.

#### 2.1.3.2. Using the Quick Search View

The DirX Directory Quick Search view displays a window from which you can perform searches on user directory entries in the DIT using simple search filters such as "begins with" or "ends with". The following figure shows an example of this view.

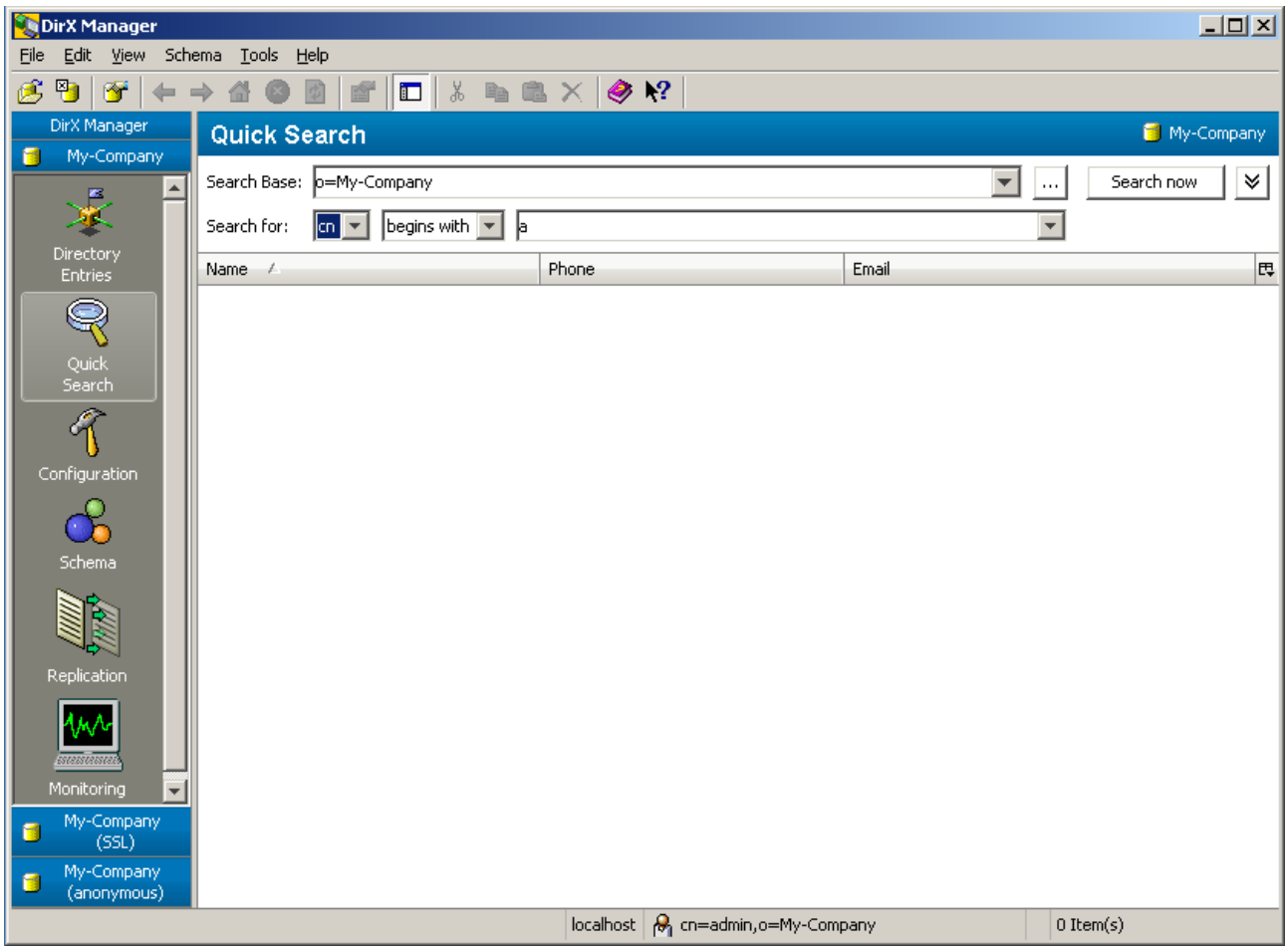



*Figure 10. DirX Directory Manager Quick Search View*

When you make a search from this view, DirX Directory Manager displays in column format in the quick search window the names of the users returned by the search and a small subset of their attributes, including their office telephone number, their mobile phone number, their email address, and the number of their department, as illustrated in the following example.

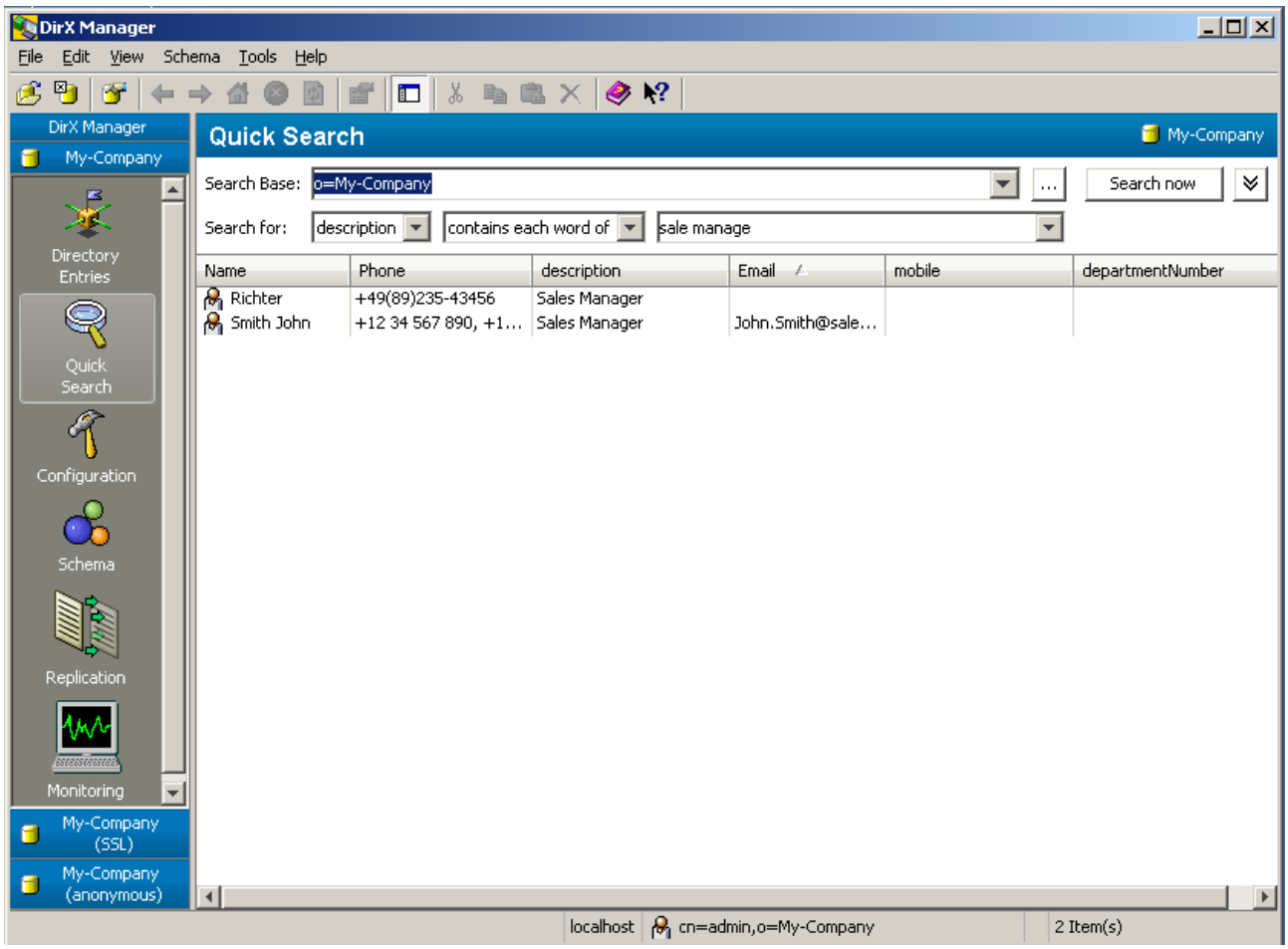


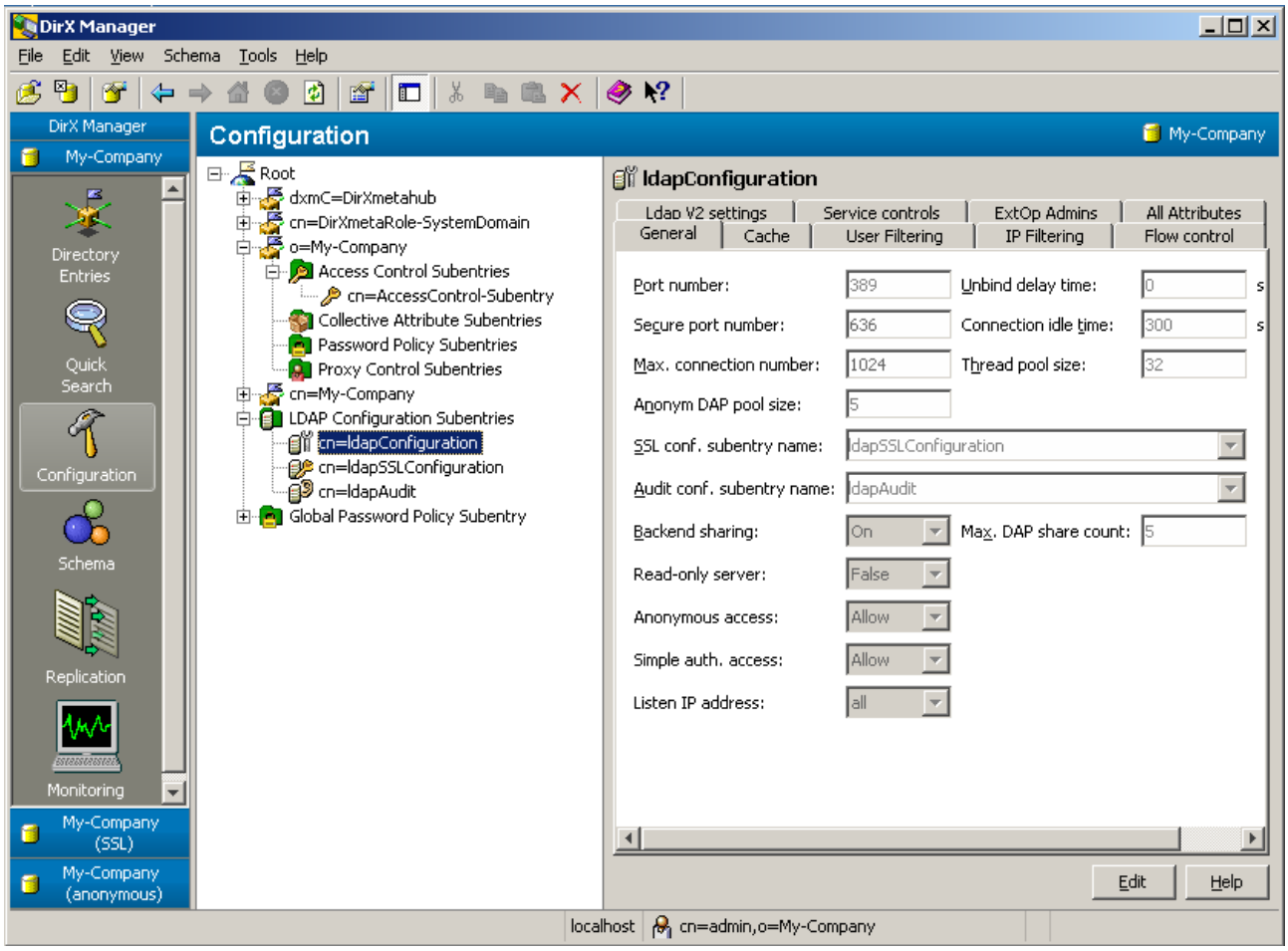

*Figure 11. DirX Directory Manager Quick Search Results*

To manage the columns that are displayed in the quick search window, right-click a column in the display and use the pop-up menu that appears.

### 2.1.3.3. Using the Configuration View

The DirX Directory Configuration view displays a hierarchical tree view of all directories in the DIT, including all administration-related entries such as access control subentries, collective attribute subentries, proxied authorization control subentries, password policy subentry, and LDAP configuration, SSL, and audit subentries. The following figure shows an example of the configuration view.

*DirX Directory Manager Configuration View*




You can use the Configuration view to:

- View and edit collective attribute subentries
- View and edit password policy subentries
- View and edit proxied authorization control subentries
- View and edit access control subentries. An ACI wizard helps you to specify new access control items
- Create and manage LDAP configuration, logging, auditing, and caching

You can also view the attributes of an LDAP server's LDAP root entry by displaying its server profile or clicking the root node in the tree.

#### 2.1.3.4. Using the Schema View

When you start DirX Directory Manager, it reads the directory schema of the LDAP server you have specified in the startup dialog. The schema view displays a hierarchical tree view of the object classes and attribute types contained in the "read-in" schema, as shown in the following example.

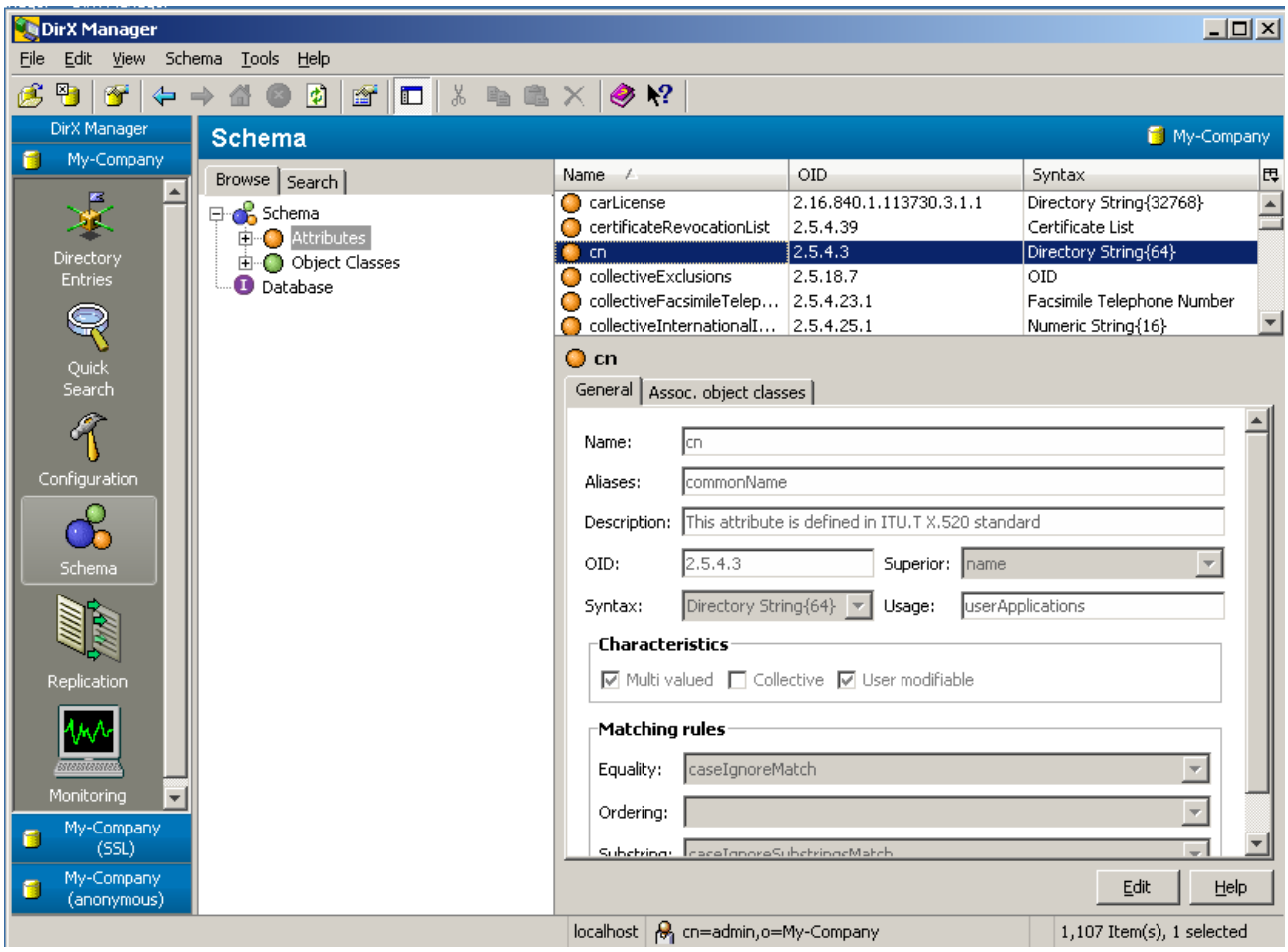



*Figure 12. DirX Directory Manager Schema View*

You can use the Schema view to:

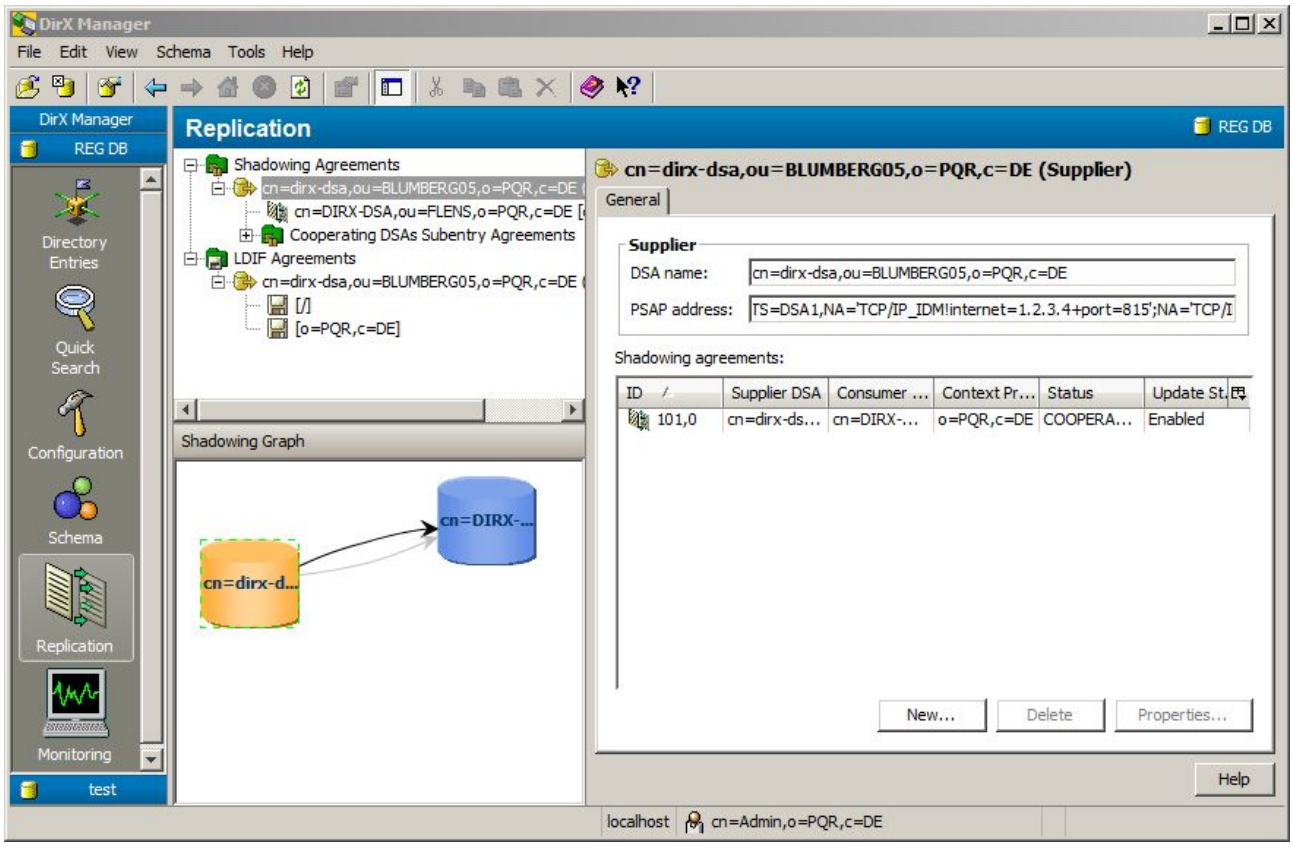
- Create new object classes and attribute types.
- View the properties of object classes and attribute types.
- Delete object classes and attribute types. Note that the delete operation is a "logical" delete: the syntax definition of the deleted object remains in the schema and you can re-create the object using an "undelete" operation.
- Compare two schemas.
- Export the complete directory schema or the differences between two schemas to an LDIF content file.
- Import all or part of a schema contained in an LDIF file into the directory schema (note that DirX Directory Manager does not perform any schema checking on the imported data). DirX Directory Manager displays a list of the object classes and attribute types contained in the file, and you can select which schema elements you want to import from this list.
- View and edit the indices maintained for attributes in the DirX Directory DSA database.

#### 2.1.3.5. Using the Replication View

The DirX Directory Replication view displays a hierarchical tree view of all shadowing and LDIF agreements. The Shadowing Graph area displays a graph of all shadowing

agreements. The following figure shows an example of the Configuration view.

*DirX Directory Manager Replication View*



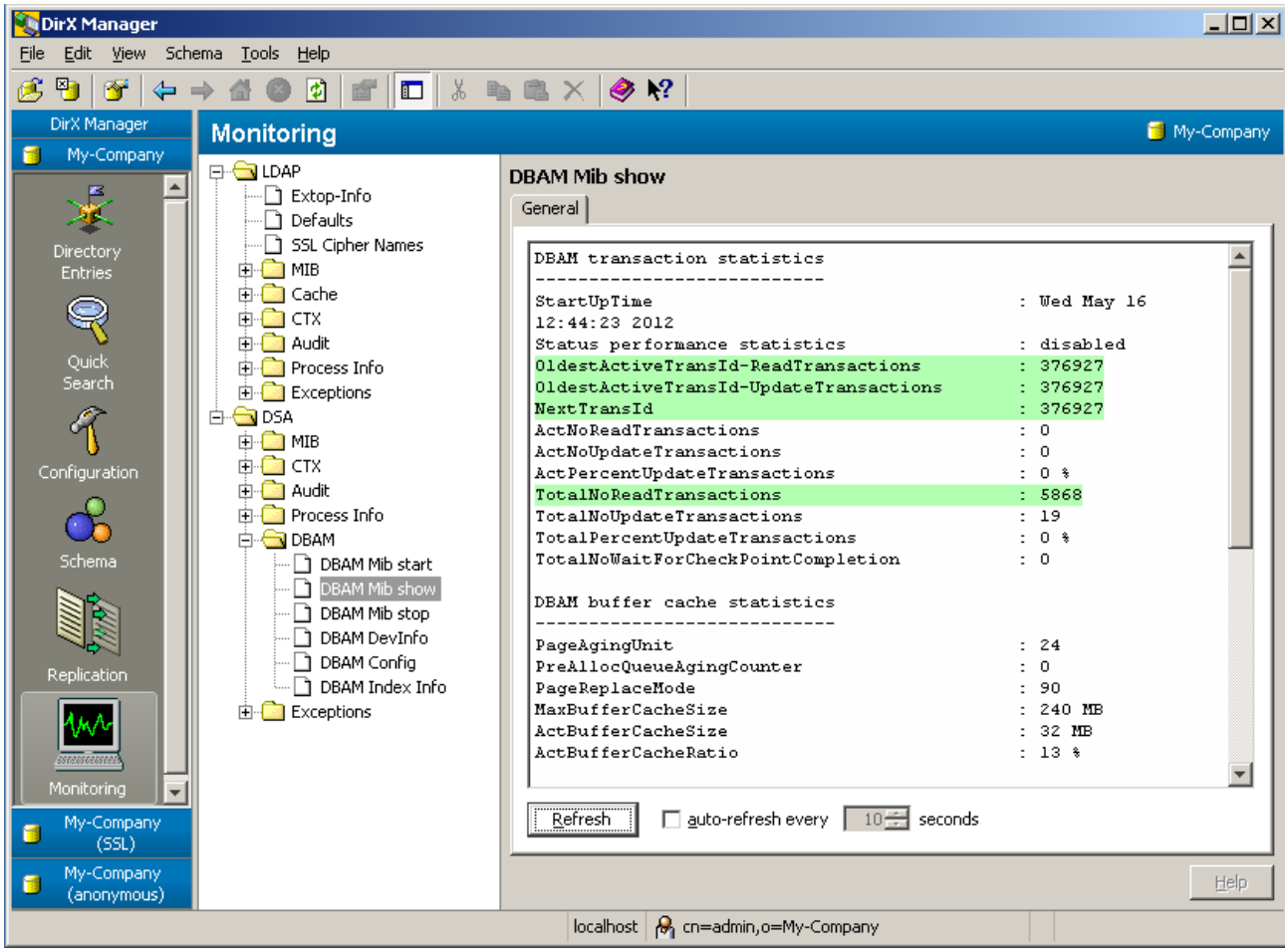

You can use the replication view to:

- Create new shadowing and LDIF agreements
- Delete shadowing and LDIF agreements
- Manage, that means establish, enable, disable or terminate, shadowing and LDIF agreements
- Display the status of shadowing and LDIF agreements
- Switch the supplier DSA of shadowing agreements

### 2.1.3.6. Using the Monitoring View

The Monitoring view displays a hierarchical tree view for the monitoring of LDAP server- and DSA-related data. For the LDAP server, a number of MIBs can be retrieved as well as information about the environment, audit and configuration default settings. From the DSA, output and detailed statistic data concerning the DBAM database are displayed. The Monitoring view is retrieved via extended LDAP operations from the server processes.

*DirX Directory Manager Monitoring View*




### 2.1.3.7. Modifying User Passwords

DirX Directory Manager provides menu selections and dialogs that permit you to change the password you use to authenticate to an LDAP server. You can also use DirX Directory Manager to create, change or delete the password of any user directory entry in the DIT, if you have the appropriate access rights to the entry.

DirX Directory Manager is a password policy-aware application; that is, it supports all management operations with respect to password expiration, aging, and account locking.

### 2.1.3.8. Adding LDAP Servers to DirX Directory Manager

When you have set up multiple DirX Directory LDAP servers (or other LDAP directories), you can make them available for management via DirX Directory Manager by adding them to DirX Directory Manager's view bar.To add an LDAP server to DirX Directory Manager's view bar, you create a server profile for it in DirX Directory Manager's Welcome view.A server profile provides DirX Directory Manager with connection and capabilities information about an LDAP server.Connection information includes the server's host name or IP address, the port on which it listens for LDAP requests, and whether or not authentication is required.Capabilities information includes the supported LDAP protocol version, whether or not the server supports SSL connections, and the name of the topmost entry in the DIT from which DirX Directory Manager is to display entries.DirX Directory Manager uses the information in a server profile during startup to connect to the correct LDAP server.DirX Directory Manager provides menu selections and dialogs to create new server profiles and

to copy, edit, and delete existing server profiles.

#### 2.1.3.9. Importing and Exporting LDAP Directory Contents

DirX Directory Manager provides menu selections and dialogs that permit you to:

- Import data from an LDIF file (content or change) or a DSML file (v1 or v2) into the LDAP server to which it is connected
- Export directory entries into an LDIF content file or a DSML v1 file

## 2.2. Using dirxcp

The **dirxcp** program is a DUA that communicates directly with DSAs over DAP and with LDAP servers over LDAP.It provides a command-line interface designed for use in scripts; directory administration with scripts provides a reliable way to perform the same task across many different servers and a method for ensuring reproducible results when compared to administration with a dialog-based tool.The *DirX Directory Administration Reference* provides a detailed description of **dirxcp** command-line syntax.

The **dirxcp** program is the primary tool for performing directory service administration and is intended for use by both system administrators and directory service end users.System administrators must use **dirxcp** over LDAP and also DAP to perform their tasks; users should only use **dirxcp** over LDAP.All **dirxcp** operations are subject to access control and schema rules: you can only perform an operation if access rights are granted to the object or the action itself is granted, and if the operation is consistent with the schema definition.

Use **dirxcp** to:

- Manage entries and subentries in the DIT, that is, manage DSEs whose DSE-type (DSET) attribute value is ENTRY or SUBENTRY.This includes the following tasks:
  - Creating subentries, including access control subentries, collective attribute subentries, and LDAP configuration subentries
  - Populating the DIT
  - Creating, modifying, and deleting entries and subentries
  - Browsing on entries in the DIT
- Manage administrative points in the DIT, that is, manage DSEs whose DSE-type attribute value is ADM_POINT. This includes the following tasks:
  - Modifying and browsing autonomous administrative points
  - Creating, modifying, browsing and deleting non-autonomous administrative points
- Manage the service control settings for directory operations; these settings control how directory operations are performed
- Return the values of the following operational attributes (operational attributes whose usage attribute value is DIRECTORY-OPERATION):
  - create-timestamp

    modify-timestamp

    - creators-name
    - modifiers-name
    - administrative-role
    - subtree-specification
    - attribute-types
    - object-classes
    - structural-object-class
    - access-control-scheme
    - prescriptive-ACI
    - subentry-ACI
    - entry-ACI

You *cannot* use **dirxcp** to:

- Manage entries whose DSE-types are not ENTRY, SUBENTRY or ADM_POINT. These entries are:
    - The root DSE
    - Glues
    - Knowledge references (subordinate, superior and cross references)
- Create autonomous administrative points
- Return the values of the following operational attributes:
    - DSE-type
    - my-access-point
    - superior-knowledge
    - specific-knowledge
    - supplier-knowledge
    - consumer-knowledge
    - secondary-shadows
    - supported-application-context
- Return information about or manage:
    - Shadowing operational bindings (stored in the cooperating-DSA attribute)
    - LDIF operational bindings (stored in the cooperating-DSA attribute)
    - User policies
    - DSA policies
    - Other global policies
- Activate and deactivate logging
- Activate, deactivate and configure auditing

- Start and stop the DSA and LDAP server
- Optimize the database configuration file

The **dirxcp** program requires that you first bind to the DirX Directory service before you can use it to perform user and administrative tasks.When you issue the **dirxcp bind** command, you can specify which protocol (DAP or LDAP) you want to use for subsequent **dirxcp** operations.You can (and should) use **dirxcp** over LDAP for all user and most administrative operations.However, some structured attributes—for example, the subtree specification (SS) attribute—and attributes with ACL syntax (prescriptive, subentry and entry ACI attributes)—are very difficult to specify in LDAP syntax.You must use **dirxcp** over DAP when you are working with these types of attributes.For example, because the SS attribute is a mandatory attribute of a subentry, you must use DAP when creating subentries.However, you can use LDAP to modify the attributes of subentries, especially the LDAP attributes of LDAP subentries.The *DirX Directory Syntaxes and Attributes* describes the DAP and LDAP syntax for DirX Directory attributes.

## 2.3. Using dirxadm

The **dirxadm** program is a DSA management tool that administrators can use to communicate directly with DSAs using an internal management API.Directory administrators must use **dirxadm** to perform directory administration operations that cannot be carried out by **dirxcp** over DAP or LDAP (because these protocols do not support the operations).The *DirX Directory Administration Reference* provides a detailed description of **dirxadm** command-line syntax.

The **dirxadm** program is a powerful program intended only for directory service system administrators.The **dirxadm** program is a DIB manipulation tool that communicates directly with the DSA using an internal management interface.It is not a DUA, cannot send LDAP or DAP requests, and cannot initiate distributed operations.

Because the **dirxadm** program can bypass access control information and schema consistency checks, it is intended for use by system administrators who have a thorough understanding of the directory information database (DIB) and its maintenance, and is not intended for end users.Use **dirxadm** only when **dirxcp** cannot perform the operation.

Use **dirxadm** to:

- Manage the entries that **dirxcp** cannot manage; these tasks include:
  - modifying the root DSE (the root DSE is created automatically during installation and cannot be deleted)
  - creating autonomous administrative points
  - creating glues
  - creating, modifying, browsing and deleting knowledge references (subordinate, superior and cross-references)
- Display the operational attributes that **dirxcp** cannot display:
  - DSE-type

    - my-access-point (this attribute is managed by the DIRX_OWN_PSAP and DIRX_DSA_NAME environment variables; see the *DirX Directory Administration Reference* for details)
    - superior-knowledge
    - specific-knowledge
    - supplier-knowledge
    - consumer-knowledge
    - supported-application-context
- Manage:
    - Shadowing operational bindings (stored in the cooperating-DSA attribute)
    - LDIF operational bindings (stored in the cooperating DSA attribute)
    - User policies
    - DSA policies
    - Other global policies
- Activate and deactivate logging
- Activate, deactivate and configure auditing
- Start and stop the DSA on Linux systems; on Windows, you can use **dirxadm** to stop the DSA and LDAP server, but you must use the Administration Tool **Services** to start it.
- Optimize the database configuration file
- Add “custom” attribute types and object classes to the standard DSA schema

System administrators using **dirxadm** should take care that they do not:

- create entries that are inconsistent with the schema in force, because clients such as **dirxcp** will not be able to access these entries
- create attributes that do not belong to the entry’s object class
- omit mandatory object class attributes
- create erroneous references
- omit operational attributes managed by the DSA

Unless you are using it to start the DirX Directory service, the **dirxadm** program requires that you first bind to the DirX Directory service before you can use it to perform administrative tasks.Unlike **dirxcp**, the **dirxadm bind** command does not offer a selection of protocols to use for subsequent **dirxcp** operations; **dirxadm** communicates with the DirX Directory service over an internal management API.

## 2.4. Using Tcl Scripts with dirxadm and dirxcp

Both **dirxcp** and **dirxadm** are built on a portable command language called the Tool Command Language (Tcl 8.3).Tcl permits the use of variables, if statements, list-processing functions, loop functions, and many other well-known command-language features.These

features allow you to create Tcl scripts to perform customized batch processing of directory administration tasks.

We strongly recommend that you develop Tcl scripts that you can use to perform complex administration tasks.Putting complex **dirxadm** and **dirxcp** commands into scripts saves you time, since you only type the commands once and avoids the hazard of making typographical errors each time you give the commands.

For example, the following script contains the **dirxadm** command necessary to establish the default password for the DSA.In the following script:

- The **dirxadm** command is identified by a text string that is displayed online with the Tcl **puts** command.
- The Tcl **catch** command is used to place the results of each command into the status variable.
- The Tcl **if** statement is used to test whether or not the command is successful.If the command fails, the **puts** command is used to display the contents of the status variable.

```
# This script is a dirxadm script that establishes
# - the default password of the DSA
#
#
# This dirxadm command modifies the default password of your
# local DSA
puts "modify own default password"
catch { modify / -addattr {DSAP={DSA={/},
                                  OPT=CHAINING,
                                  AP={PWD=MY_NEW_PASSWORD}
                                 }
                           }
      } status

if {$status == ""}   \
   then {puts "operation ok"}   \
   else {puts "$status"}
```

Note the presence of the line continuation character “\”.In scripts it is needed if Tcl commands stretch over several lines and the line break is not enclosed in curly braces { }.In interactive mode, you cannot use the line continuation character when you want to continue a **dirxcp** command on a new line.You can issue a **dirxcp** command that is longer than will fit on a line simply by continuing to type the command line without typing an ENTER or RETURN.Although the resulting command line will appear to be “wrapped” on the display, internally it is a single line.

You can find more examples of Tcl scripts in the directory *install_path***/scripts**, which is

installed on your system when you install DirX Directory.Refer to a book about Tcl for instructions on developing Tcl scripts; for example, *Practical Programming in Tcl and Tk*, by Brent B. Welch, or *Tcl and the Tk Toolkit*, by John K. Ousterhout.

## 2.5. Using dirxload

The **dirxload** program is a command-line tool that you can use to bulk-load very large amounts of data very quickly into the DirX Directory database.The tool can load 1,000 entries per second—one million entries in twenty minutes.

Before you run **dirxload**, Meta The data to be loaded must be contained in one or more LDIF content files; you specify the pathnames of these files on the **dirxload** command line.

You can run **dirxload** in "simulation" mode, where it processes the content file(s), but does not actually create the data in the DirX Directory database.You can also specify a **dirxload** command line option that saves LDIF entries that **dirxload** rejects into a file, which allows you to evaluate and fix any problems in the LDIF content file(s) before you perform the actual loading operation.

See the *DirX Directory Administration Reference* for complete details about **dirxload** command-line syntax.

## 2.6. Using dirxmodify

The **dirxmodify** program is a command-line tool that permits you to load LDIF content and LDIF change files into any LDAP directory—not just DirX Directory—over LDAP v3. You can use **dirxmodify** to:

- Load an empty LDAP directory with entries in an LDIF content file
- Synchronize an LDAP directory with directory entry modifications in an LDIF change file
- Update an LDAP server schema with modifications in an LDIF change file
- Perform off-line processing and checking of LDIF files

When you invoke **dirxmodify**, it binds to an LDAP server; you can use command line options to specify the LDAP server to which you want to connect.You can bind as an anonymous user or you can provide simple authentication credentials (distinguished name and password) as command-line options.The user represented by the distinguished name must have the permissions that are necessary to perform modifications to the target directory.

You can run **dirxmodify** in "simulation" mode, where it processes the content or change file(s), but does not actually create or modify the data in the DirX Directory database.You can also specify a **dirxmodify** command line option that saves LDIF entries that **dirxmodify** rejects into a file, which allows you to evaluate and fix any problems in the LDIF content file(s) before you perform the actual load operation.

The **dirxmodify** tool provides command line options that allow you to make changes to the LDIF content or change files "on the fly" during the loading process.The tool makes the

updates internally to the read-in files; the files themselves are not changed.You can also specify attributes in the files that should not be loaded into the database, and you can use a command line option to position **dirxmodify** so that it reads from a specific entry in a content or change file.

The **dirxmodify** program performs its directory loading operations over LDAP.As a result, its bulk-loading operation is slower than **dirxload**, which can perform bulk loading at 10 to 20 times the speed of a protocol-initiated operation.

See the *DirX Directory Administration Reference* for complete details about **dirxmodify** command-line syntax.

## 2.7. Using dirxbackup

The **dirxbackup** program is a command-line tool that you can use to perform normal daily backup operations on the DirX Directory database in conjunction with file compress/uncompress tools such as **gzip**.The **dirxbackup** program allows you to:

- Save the active DirX Directory (DBAM) database to a database archive file (or to standard output)
- Restore a saved DirX Directory database in an archive file (or from standard input) to the active DirX Directory database
- Verify a database archive file for consistency

Do not use non-DirX backup tools to back up the DBAM database.Only **dirxbackup** performs the necessary synchronization with other DirX processes using the database.Using non-DirX backup tools that access DBAM database files or devices can produce inconsistencies in the database and / or in the dirxbackup archive file.

See the *DirX Directory Administration Reference* for complete details about **dirxbackup** command-line syntax and operation.

# 3. Getting Started with DirX Directory Setup

Setting up the DirX Directory service consists of the following basic tasks:

1. Configuring the disk(s) for the DBAM database
2. Installing the DirX Directory service
3. Creating a profile for the DBAM database
4. Initializing the DBAM database
5. Loading the directory data into the DBAM database
6. Starting the service

This chapter takes you through a simple example setup scenario that introduces you to each task. The chapter provides a setup scenario for Windows Server and a setup scenario for Linux.

## 3.1. General Notes

DirX Directory allows you to use the directory service in a distributed environment.Several systems, each running a DSA, are connected for the purpose of shadowing parts of the DIT or distributing the DIT or both.

In a distributed environment, the system clocks on the different systems must be well synchronized.This can be achieved, for example, by using a time synchronization service on each system.If the system clocks are not well synchronized, it is possible that DSP or DISP binds using the simple protected authentication method can fail because credentials have become invalid.Note, however, that you can increase the credentials timeout limits to alleviate this effect.

## 3.2. Setting up the Sample Service on Windows

This section describes how to set up an example DirX Directory service on Windows.In the example scenario for Windows:

- The target machine has two disks: Disk 0 is a populated disk.Disk 1 is an empty disk, which is 36GB and is configured as a basic disk.
- We will use one directory data device to hold all of the DirX directory data types (there is always only one transaction device).
- We will load sample directory data supplied in an LDIF file that is provided with the DirX Directory installation.

### 3.2.1. Configuring the Disks for the DBAM Database

On Windows systems, configuring disks for the DBAM database consists of two steps:

1. Optionally upgrading any basic disks to be used to store the database to dynamic disks. (See Windows Server Disk Management documentation to get more information on this topic.)
2. Creating the volumes on the disk(s) that will be used as raw devices for the DBAM directory data and transaction devices

We use the MMC Disk Management snap-in to perform these tasks.

#### 3.2.1.1. Upgrading the Target Disk

In our example, we will use Disk 1 to contain the DBAM database. So, we need to start the MMC Disk Management snap-in and upgrade this disk:

1. Right-click the **Start** button.
2. Click **Computer Management**.
3. Click the Disk Management folder to start the Disk Management snap-in, as shown in the following figure.

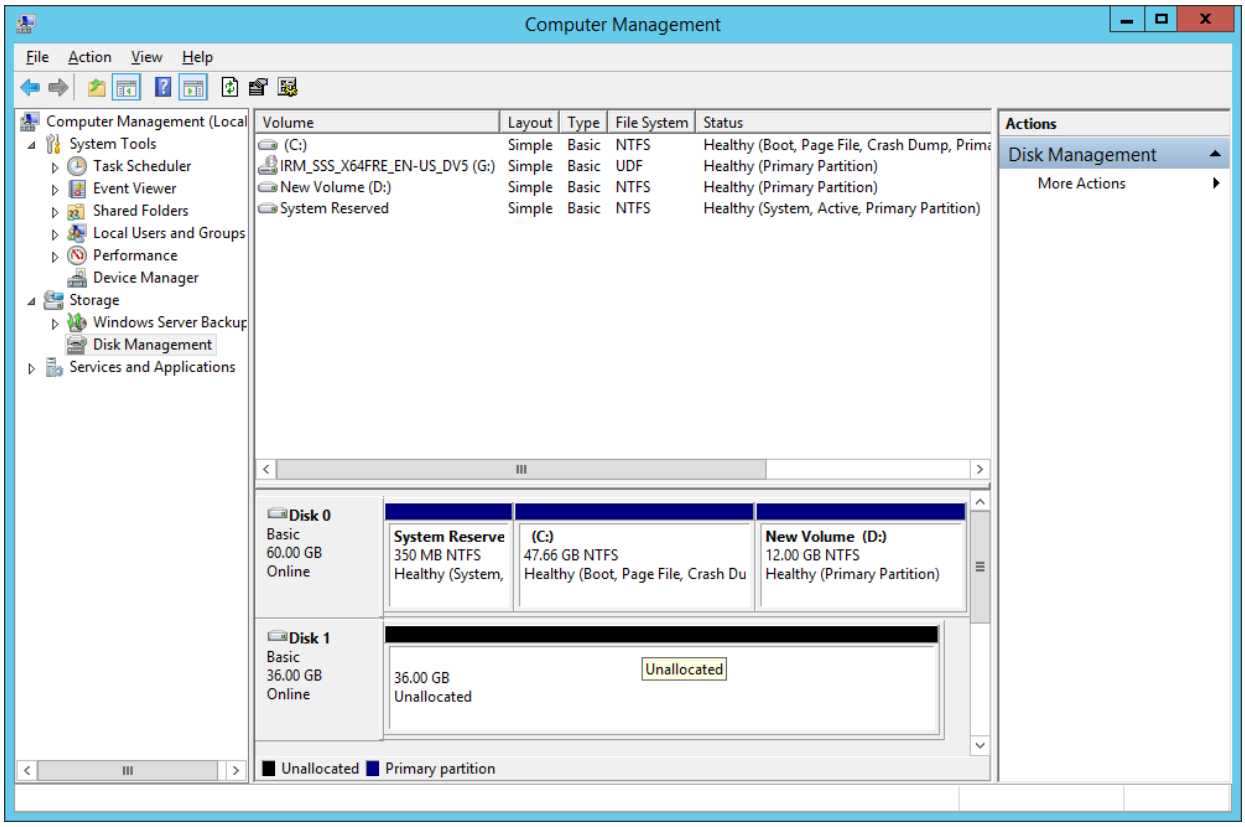



*Figure 13. Disk Management Snap-In on Windows Server*

4. In the left-hand window of the Disk Management display, right-click **Disk 1** and select **Upgrade to Dynamic Disk**. The service displays the following dialog:

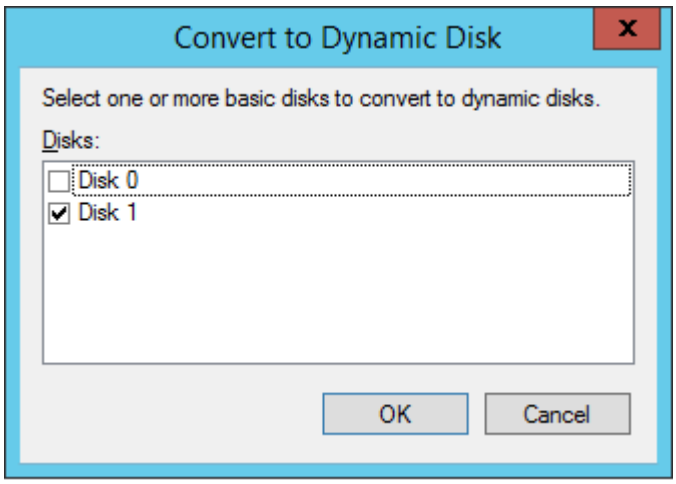



*Figure 14. Upgrade to Dynamic Disk Dialog*

5. Check **Disk 1**, then click **OK**.

Now we have converted Disk 1 from a basic to a dynamic disk.

#### 3.2.1.2. Creating the Volumes

Our example uses the simplest DBAM configuration: one volume for directory data and one volume for transaction data. So, we next use the snap-in Create Volume Wizard to split Disk 1 into two volumes: one for the directory data device and one for the transaction device. We will create both these volumes as simple volumes (the section “Dynamic Disks” in the chapter “Understanding DBAM and Storage Management for an explanation of the different Windows dynamic disk types).

Beforehand, however, we use the Windows file system to create two folders on **C:**:

- **C:\DirXstorage\data**
- **C:\DirXstorage\translog**

We will use these folders later on for mounting the volumes.

Next, we use the snap-in to create the volume for the directory data device:

1. Right-click **Disk 1** and select **New Simple Volume**. This action starts the New Simple Volume Wizard.

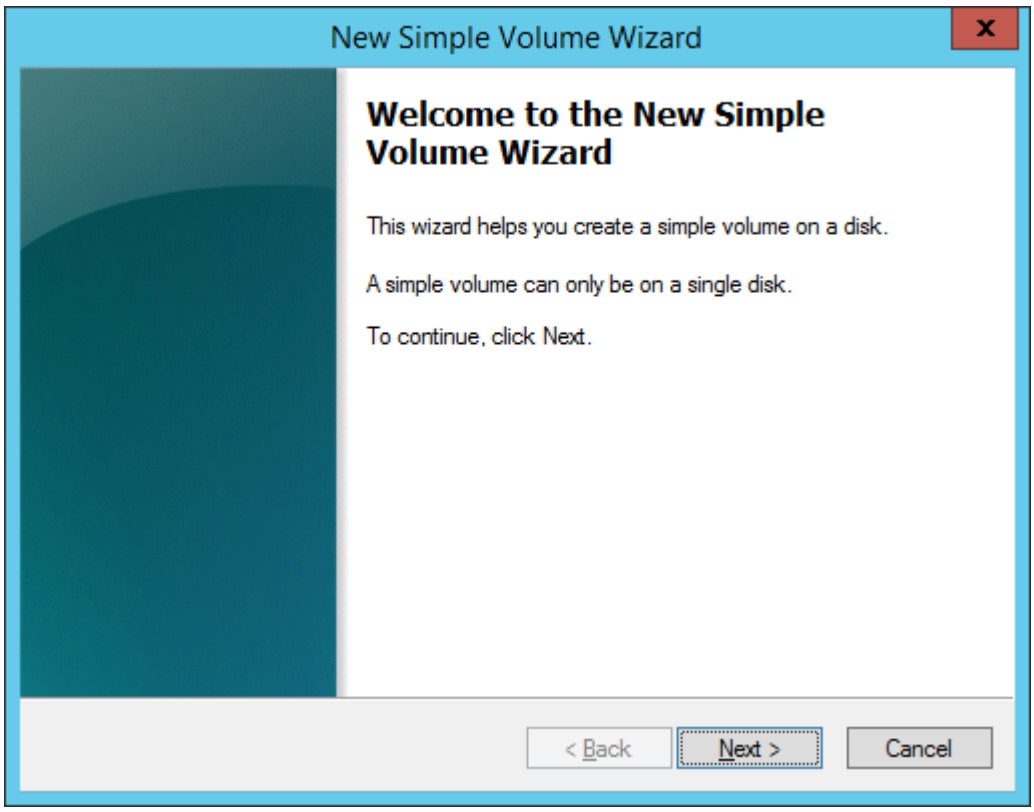



*Figure 15. New Simple Volume Wizard*

2. The wizard displays the Select Disks dialog, where you can set the disk size for the volume. In **Size for selected disk**, select **32768** MB for the volume's disk size; this amount of space can accommodate a directory service configuration of 13 million directory entries and 20 attribute indexes. After you have made this selection, click **Next**.

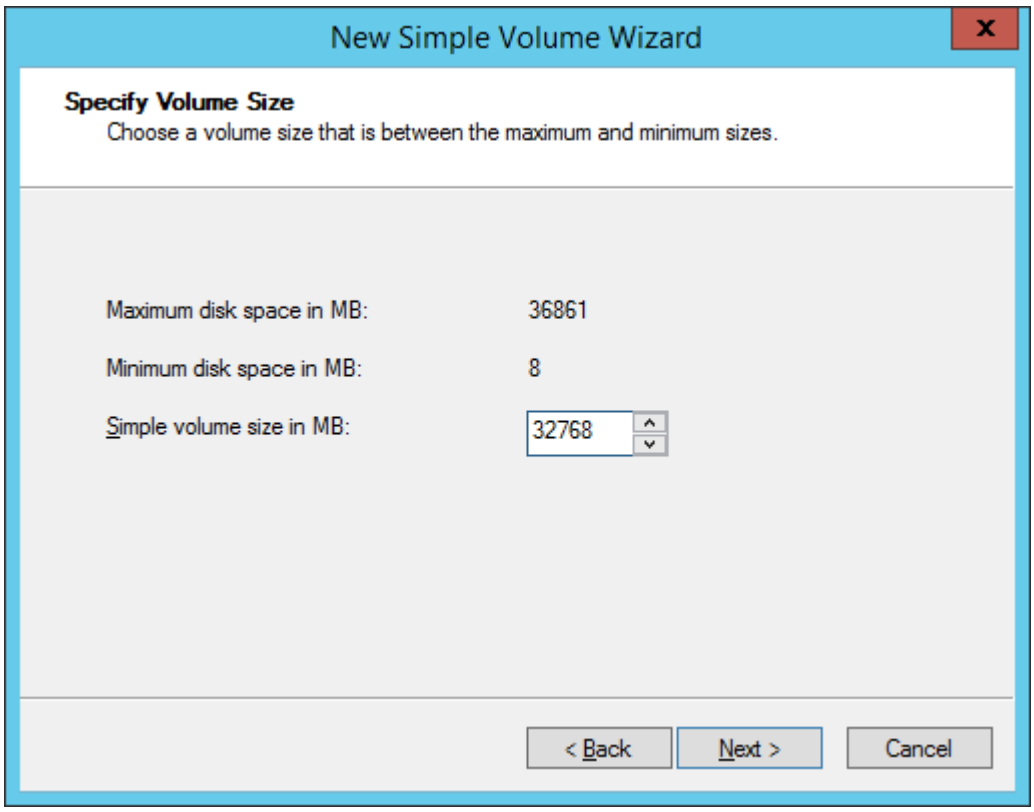



*Figure 16. Select Disks Dialog*

3. The Wizard displays the Assign Drive Letter or Path dialog. Click **Mount this volume at an empty folder that supports drive paths** and browse to the directory **C:\DirXstorage\data** that we created earlier. Select this folder, then click **Next**.

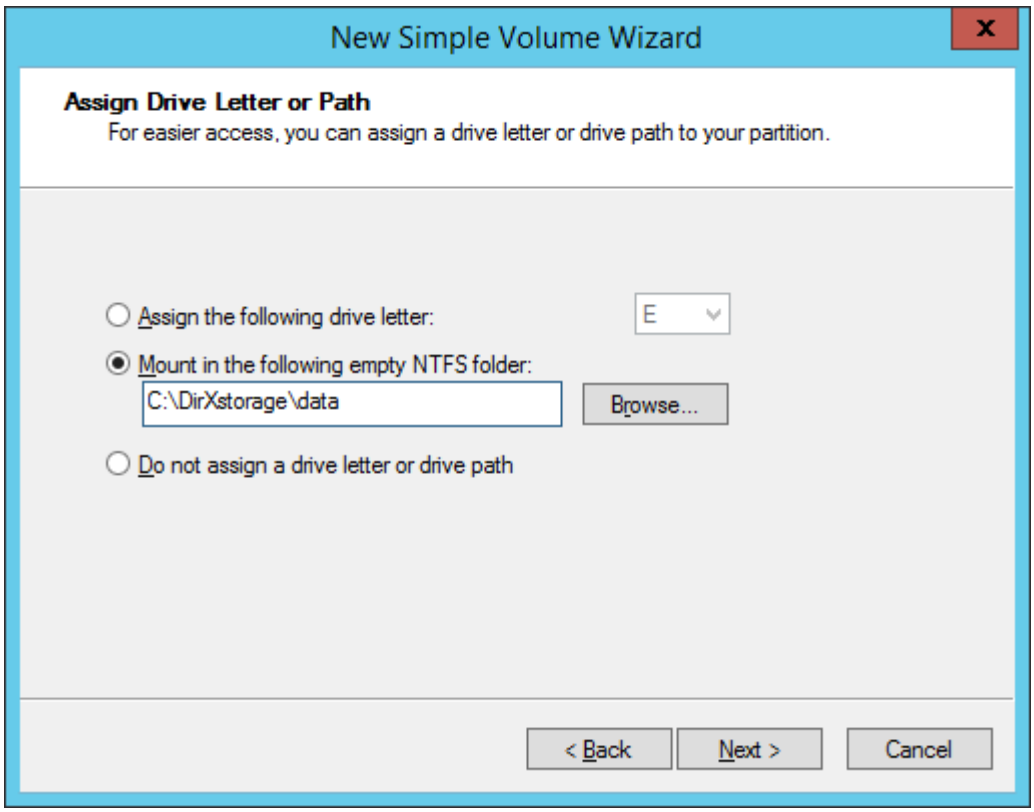



*Figure 17. Assign Drive Letter or Path Dialog*

4. The Wizard displays the Format Volume dialog. Click **Do not format this volume**, then click **Next**.

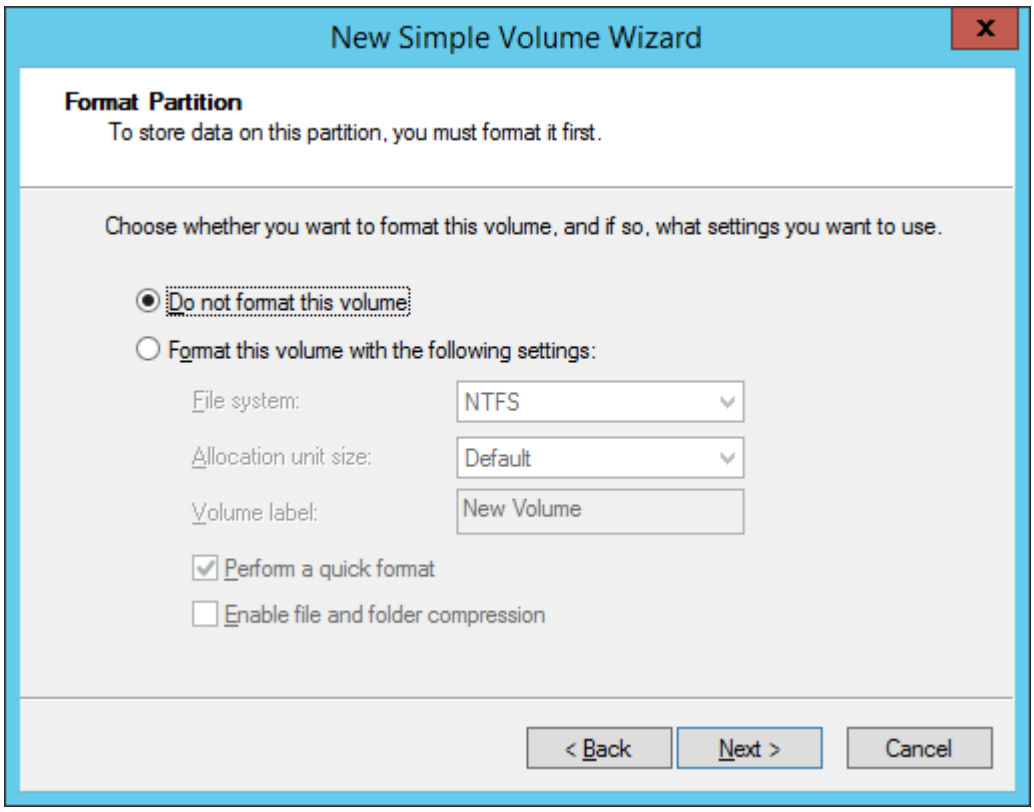



*Figure 18. Format Volume Dialog*

5. The Wizard displays the volume settings you have selected. Review the settings to make sure they're correct, then click **Finish**.

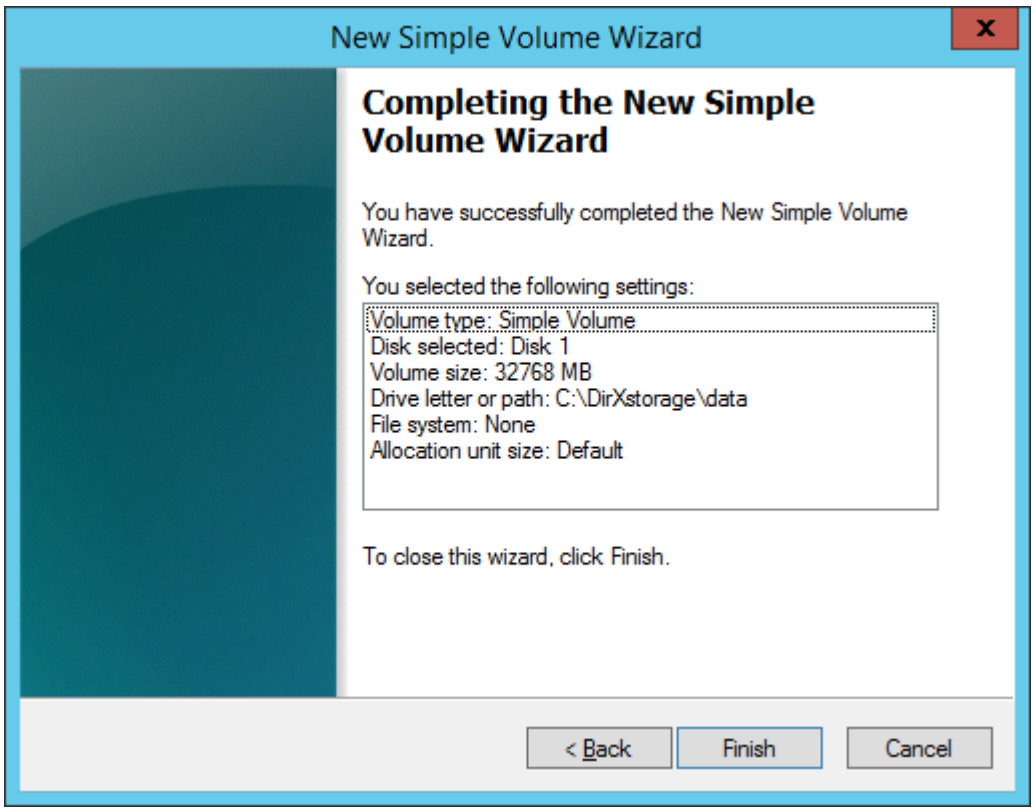



*Figure 19. Volume Wizard Completion Dialog*

Now we have created a Windows volume—in the DBAM database model, a raw device—for the directory data device. Next, we create another volume for the transaction device:

1. Right-click Disk 1 and select **Create Volume**. This action starts the Create Volume Wizard. Click **Next**.
2. In the Select Volume Type dialog, check **Simple Volume**, then click **Next**.
3. In the Select Disks dialog, the wizard displays the remaining amount of space left on Disk 1 in **Size for selected disk**. Use this size and click **Next**.
4. In the Assign Drive Letter or Path dialog, select **Mount this volume at an empty folder that supports drive paths** and browse to the directory **C:\DirXstorage\translog** that we created earlier. Select this folder, then click **Next**.
5. In the Format Volume dialog, click **Do not format**, then click **Next**.
6. The Wizard displays the volume settings you have selected. Review the settings to make sure they're correct, then click **Finish**.

Now we have completed the configuration of Disk 1 for the DBAM data and transaction devices.

### 3.2.2. Installing DirX Directory

The next step is to install the DirX Directory software from the product CD. Follow the instructions given by the Setup Wizard. In our example, we install the software into the directory **C:\Program Files\DirX\Directory**. The DirX Directory installation requires approximately 100MB. You should also ensure that there is enough space to accommodate any files you create during your use of the directory service; for example, log files, LDIF

content files, and LDIF change files. The amount of space you'll need depends on how you plan to use the example set-up.

### 3.2.3. Creating a Profile for the DBAM Database

Now that we have created our volumes (DBAM raw devices) and installed the DirX Directory software, we next use the DirX Directory administration tool **dbamconfig** to create a profile for the DBAM database. The profile links the volumes we created earlier to the DBAM data device/transaction device format. To create the profile:

1. Start the Command Prompt window and run the following **dbamconfig** command:

   ```
   dbamconfig
       -c -D"C:\DirXstorage\data"
       -T"C:\DirXstorage\translog"
       -Pprofile1
   ```

   We use the double quotation marks to ensure that **dbamconfig** (and the other DirX Directory administrative commands used in this example) will be able to parse any space characters that may occur in the pathname (although there are none in this example).

   The command should return the status:

   ```
   Creation of profile 'profile1' was successful
   ```

2. Use the following **dbamconfig** command to display the profile you just created:

   ```
   dbamconfig -sl
   ```

The command should display the profile, for example:

+

```
ProfileName: profile1, ID=3
  DataDevice(s):
    Mountpoint: C:\DirXstorage\data\
    VolumeName: \\?\Volume{9fc2a96c-a374-11e3-80cb-000c293728b5}
    DeviceSize: 32.000 GB
      Logical device(s):
        BlockType(s): GENERAL | BITSTR | PSEUDO | TREE
        DeviceSize:   6.400 GB
```

```
          BlockType(s): REAL
          DeviceSize:   12.800 GB

          BlockType(s): AVIDX
          DeviceSize:   12.800 GB

    TransactionDevice:
      Mountpoint: C:\DirXstorage\translog\
      VolumeName: \\?\Volume{9fc2a9a8-a374-11e3-80cb-000c293728b5}
```

When we used the **dbamconfig** command to create the profile, we did not specify how the space on the directory data device was to be allocated to the different data types. As a result, **dbamconfig** used a default allocation: 40% of the device to real directory data, 40% to attribute index data, and the remainder to the other types of data.

### 3.2.4. Initializing the DBAM Database

The next step is to use the **dbamboot** command to initialize an empty database that corresponds to the database profile **profile1** that we just created with **dbamconfig**.

From the Command Prompt, run the following **dbamboot** command

```
dbamboot -Pprofile1
```

The command should return the message:

```
DB successfully initialized
        DB-limits:
                13418476 objects
                      20 attribute index(es)
```

Here you can see that the database you've initialized can accommodate approximately 12 million entries and 20 attribute index types.

You can use the **dbamdevinfo** command to view the empty database:

```
dbamdevinfo

DSA profile:               profile1, ID=1
Real object block size:    1 KB
AVIDX cluster size:        32 MB
```

```
Maximum number of indices: 880

Logical device:  GENERAL | BITSTR | PSEUDO | TREE
    In use:         0.00 %  (  0.130 MB of    6.398 GB)
    Fragmentation:  0.00 %
Logical device:  REAL
    In use:         0.00 %  (  0.000 MB of   12.797 GB)
    Fragmentation:  0.00 %
Logical device:  AVIDX
    In use:         0.00 %  (  0.000 MB of   12.800 GB)
    Fragmentation:  0.00 %

Attribute index specific device info:

        Number of indices: 0      (maximum 80)
        Cluster usage:     0.00 % (  0 of 895)
```

Here you can see how the default allocation for the different directory types has been applied to Disk 1.

### 3.2.5. Loading the Directory Data

Now we are ready to load directory data into the empty database. In this example, we use the **dirxload** administration tool to populate the empty database with sample directory data in an LDIF content file named **Complete_DB.ldif**, which is supplied with the DirX Directory installation in *install_path***scripts\stand_alone\default**.

1. In the Command Prompt window, change directory to **C:\Program Files\DirX\Directory\scripts\stand_alone\default**.
2. Use the following **dirxload** command to load the example database **Complete_DB.ldif** into the DBAM database:

```
dirxload -f Complete_DB.ldif

Attribute objectClass(0) INITIAL indexed, file
C:\PROGRA~1\DirX\Directory\tmp\0-INITIAL-1796.iaf
Attribute cn(3) INITIAL indexed, file
C:\PROGRA~1\DirX\Directory\tmp\3-INITIAL-1796.iaf
Attribute cn(3) FINAL indexed, file C:\PROGRA~1\DirX\Directory\tmp\3-
FINAL-1796.iaf
Attribute sn(4) INITIAL indexed, file
C:\PROGRA~1\DirX\Directory\tmp\4-INITIAL-1796.iaf
Attribute sn(4) FINAL indexed, file C:\PROGRA~1\DirX\Directory\tmp\4-
```

```
FINAL-1796.iaf
Attribute c(6) INITIAL indexed, file
C:\PROGRA~1\DirX\Directory\tmp\6-INITIAL-1796.iaf
Attribute c(6) FINAL indexed, file C:\PROGRA~1\DirX\Directory\tmp\6-
FINAL-1796.iaf
Attribute o(13) INITIAL indexed, file
C:\PROGRA~1\DirX\Directory\tmp\13-INITIAL-1796.iaf
Attribute o(13) FINAL indexed, file
C:\PROGRA~1\DirX\Directory\tmp\13-FINAL-1796.iaf
Attribute collectiveOrganizationName(14) INITIAL indexed, file
C:\PROGRA~1\DirX\Directory\tmp\14-INITIAL-1796.iaf
Attribute collectiveOrganizationName(14) FINAL indexed, file
C:\PROGRA~1\DirX\Directory\tmp\14-FINAL-1796.iaf
Attribute ou(15) INITIAL indexed, file
C:\PROGRA~1\DirX\Directory\tmp\15-INITIAL-1796.iaf
Attribute ou(15) FINAL indexed, file
C:\PROGRA~1\DirX\Directory\tmp\15-FINAL-1796.iaf
Attribute collectiveOrganizationalUnitName(16) INITIAL indexed, file
C:\PROGRA~1\DirX\Directory\tmp\16-INITIAL-1796.iaf
Attribute collectiveOrganizationalUnitName(16) FINAL indexed, file
C:\PROGRA~1\DirX\Directory\tmp\16-FINAL-1796.iaf
1025 entries created
0 entries rejected
```

Step 1 is optional: you can supply the full path to the example database to **dirxload**, as follows:

```
dirxload -f"C:\Program
Files\DirX\Directory\scripts\stand_alone\default\Complet
e_DB.ldif"
```

Now we have configured and initialized the DBAM database and have loaded directory content into it. We can then start the service and use any LDAP client to view the data. For example, we can use the command-line tool **dirxcp** to view the data in text format, or we can use DirX Directory Manager to view the data in a GUI format.

### 3.2.6. Starting the Service

To start the DirX Directory service use the Windows Administration Tool **Services**:

1. Start the **Services** Administration Tool.

2. Right-click **DirX Service**, then click **Start**.

Now the service is running.

### 3.2.7. Viewing the Directory Data with dirxcp

To use **dirxcp** to view the directory data you loaded into the DBAM database:

1. In the Command Prompt window, start **dirxcp**:

```
dirxcp
```

2. The **dirxcp** program requires that you first bind to the DirX Directory service before you can perform any further **dirxcp** operations. Use the following **dirxcp** operation to make an unauthenticated (anonymous) bind to the service:

```
dirxcp> bind
```

3. Now you can use **dirxcp** to examine the database. For example, you can use the following **dirxcp** operation to search the database:

```
dirxcp++>++ search / -sub -p
```

```
 1) /O_DUTF8=My-Company/CN_DUTF8=admin
 2) /O_DUTF8=My-Company/OU_DUTF8=Sales
 3) /O_DUTF8=My-Company/OU_DUTF8=Development
 4) /O_DUTF8=My-Company/OU_DUTF8=Accounting
 5) /O_DUTF8=My-Company/OU_DUTF8=Product Development
 6) /O_DUTF8=My-Company/OU_DUTF8=Product Testing
 7) /O_DUTF8=My-Company/OU_DUTF8=Human Resources
 8) /O_DUTF8=My-Company/OU_DUTF8=Payroll
 9) /O_DUTF8=My-Company/OU_DUTF8=Sales/CN_DUTF8=Smith John
10) /O_DUTF8=My-Company/OU_DUTF8=Sales/CN_DUTF8=Mayer
11) /O_DUTF8=My-Company/OU_DUTF8=Sales/CN_DUTF8=Hohner
12) /O_DUTF8=My-Company/OU_DUTF8=Sales/CN_DUTF8=Richter
13) /O_DUTF8=My-Company/OU_DUTF8=Sales/CN_DUTF8=Abele
14) /O_DUTF8=My-Company/OU_DUTF8=Sales/CN_DUTF8=Reichel
15) /O_DUTF8=My-Company/OU_DUTF8=Development/CN_DUTF8=Digger
16) /O_DUTF8=My-Company/OU_DUTF8=Development/CN_DUTF8=Filler
17) /O_DUTF8=My-Company/OU_DUTF8=Development/CN_DUTF8=Tinker
18) /O_DUTF8=My-Company/OU_DUTF8=Development/CN_DUTF8=Morton
```

```
19) /O_DUTF8=My-Company/OU_DUTF8=Accounting/CN_DUTF8=Dasya
Linkletter
20) /O_DUTF8=My-Company/OU_DUTF8=Accounting/CN_DUTF8=Antonio
Tymchuk
21) /O_DUTF8=My-Company/OU_DUTF8=Accounting/CN_DUTF8=Talia
McKinnon
22) /O_DUTF8=My-Company/OU_DUTF8=Accounting/CN_DUTF8=Becky
McCready
23) /O_DUTF8=My-Company/OU_DUTF8=Accounting/CN_DUTF8=Corly
Wheatley
24) /O_DUTF8=My-Company/OU_DUTF8=Accounting/CN_DUTF8=Lilin Markham
:
```

The colon ( : ) at the end of the list indicates that there is more data to display (enter “q[uit]” to discard further output).

### 3.2.8. Viewing the Directory Data using DirX Directory Manager

You can view the DirX Directory database from any LDAP client. In this example, we use DirX Directory Manager

To start DirX Directory Manager:

1. Click **Start**.
2. Search for **DirX Directory Manager**, and then click **DirX Directory Manager**.
3. DirX Directory Manager displays its Welcome dialog. Click the **My-Company (SSL)** view group, then click **Open** and specify credentials or bind as anonymous.

In the My-Company (SSL) view group, click the **Directory Entries** view. DirX Directory Manager displays a view pane from which you can browse and search for entries in the DirX Directory database. The following figure shows the Browse tab.

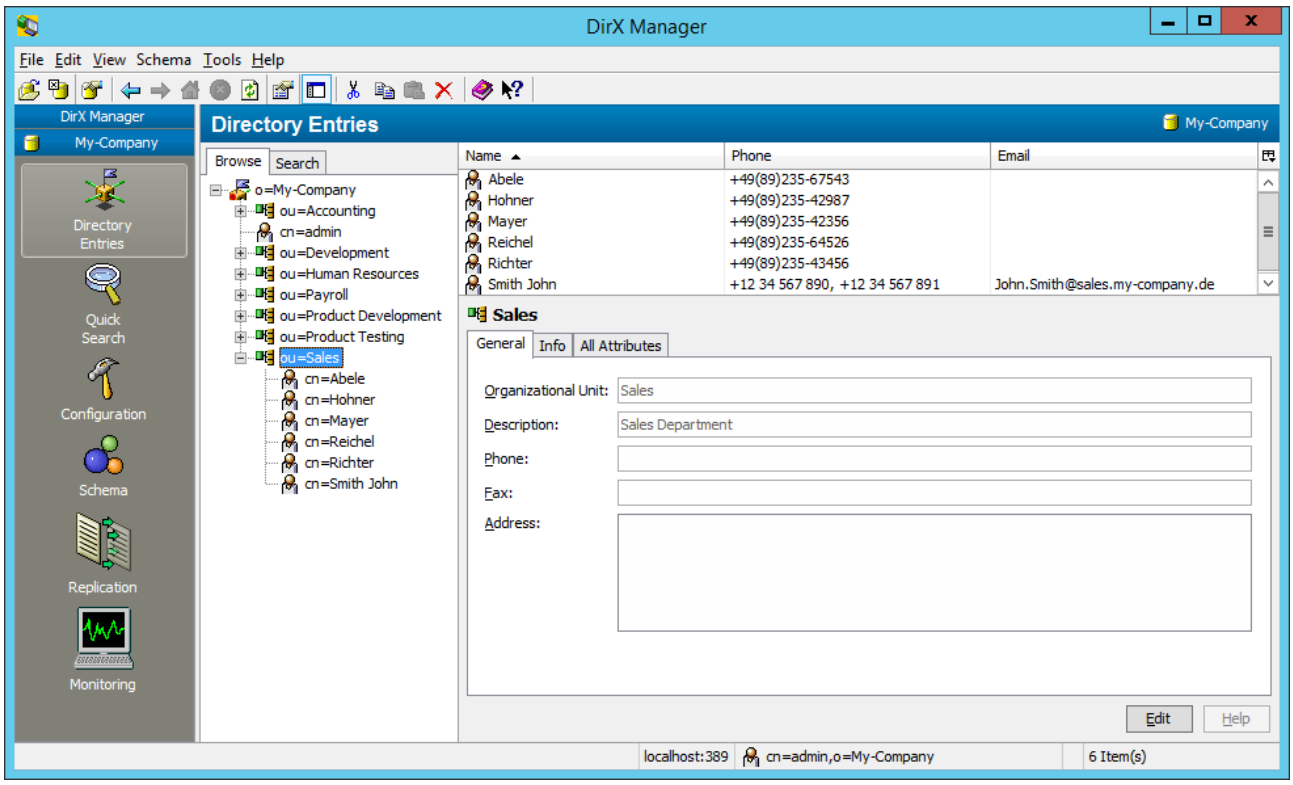



*Figure 20. DirX Directory Manager Example Directory Browse*

In this example, we search the whole subtree **o=My-Company** for directory entries whose Object Class is Person and whose Common Names begin with “A” and whose Description begins with “S”. The following figure illustrates the result.

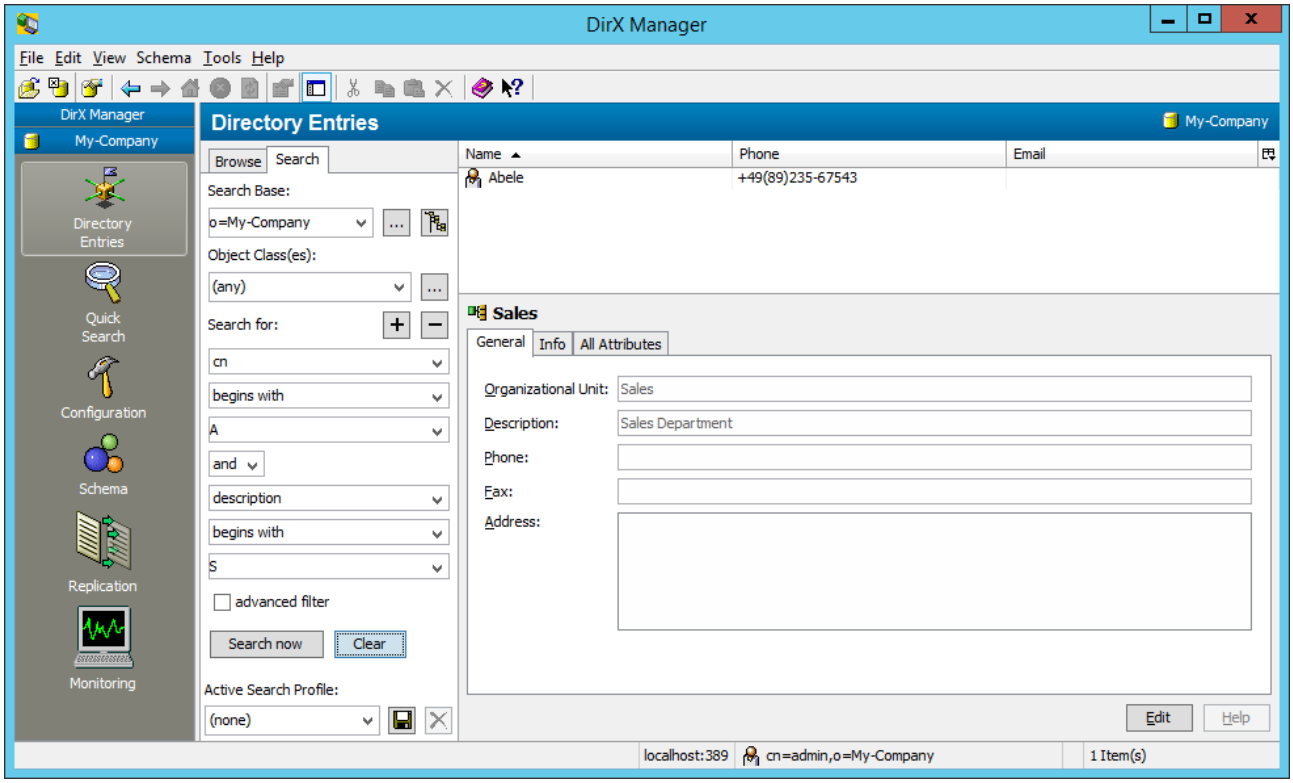



*Figure 21. DirX Directory Manager Example Directory Search 1*

In this example, we search for directory entries whose Surname (sn) attribute begins with

the string “sm”. Initially, the search base is supposed to be “o=My-Company”.

To view or edit the search base, click .

The following figure illustrates the result.

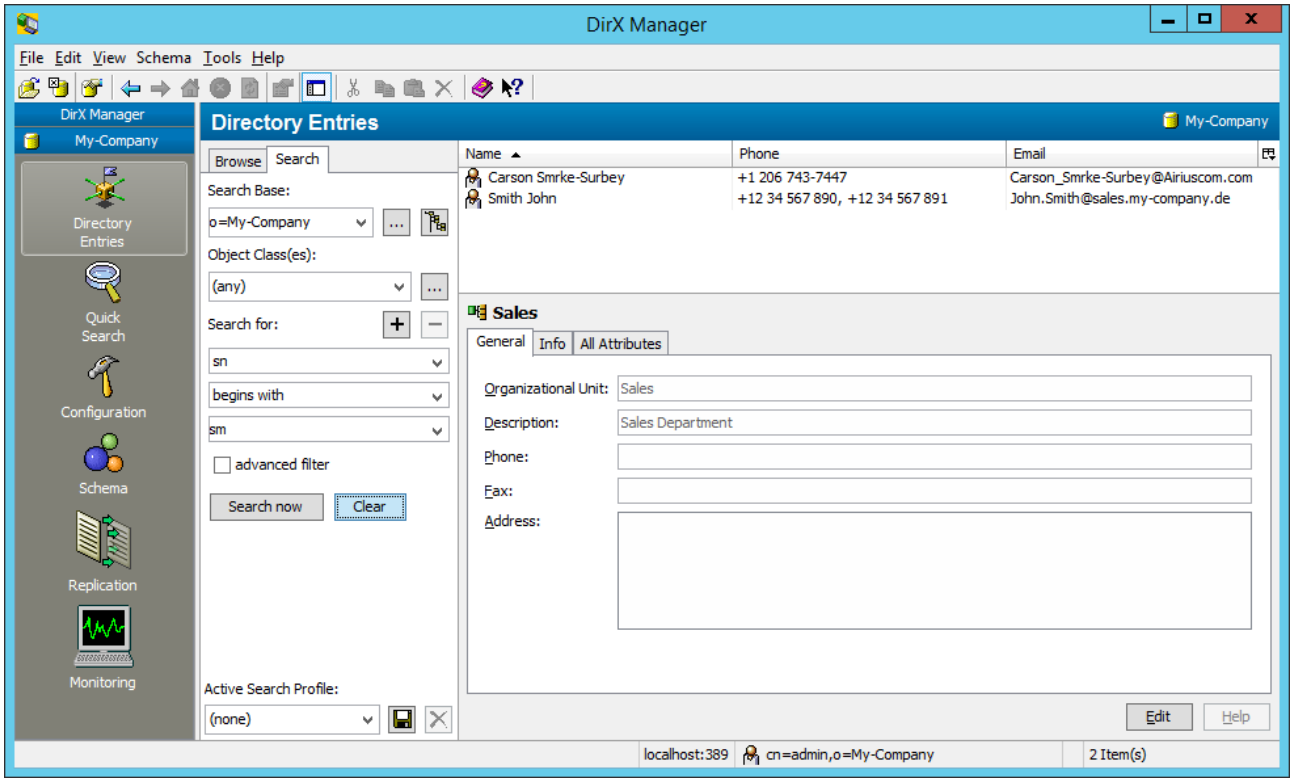



*Figure 22. DirX Directory Manager Example Directory Search 2*

### 3.2.9. Making Changes to the Example

If you want to re-load the example database or load your own directory data, you can use **dbamboot** to reinitialize the DBAM database. For example, the command:

```
dbamboot -Pprofile1
```

deletes all the data you loaded with **dirxload**.You can run **dirxload** again to re-load **CompleteDB.ldif** or another LDIF content file of your choice.

You can use **dbamconfig** to delete the database profile you created as long as it is not the active profile.For example, the command:

```
dbamconfig -d -Pprofile1
```

deletes the **profile1** profile that we created earlier.

# 3.3. Setting up the Sample Service on Linux

This section describes how to set up an example DirX Directory service on Linux.In the example scenario:

- The target machine has three disk groups of two physical disks each, managed by a hardware or software RAID system.The first disk group holds the root (system).The other two disk groups have been reserved for use as DBAM storage devices.
- We will use one disk group to hold all of the DirX directory data types and the other disk group to hold the transaction data.We will use both disks exclusively for DBAM storage.
- We will load sample directory data supplied in an LDIF file that is provided with the DirX Directory installation.

## 3.3.1. Configuring the Disks for the DBAM Database

On Linux systems, configuring disks for the DBAM database consists of the following steps:

1. Creating a group definition in the system for DirX Directory administration and adding a new user as a member of this group
2. Assigning symbolic links to the raw data slices or “raw devices” to be used for the DBAM directory data and transaction devices
3. Changing the permissions on the DBAM raw devices so that users in the DirX group can access them

You must be logged in as user **root** (superuser) to perform all of these steps.

#### 3.3.1.1. Creating the DirX Group and User

In this procedure, you first create a new group that defines users who will perform DirX Directory administration tasks. You then create a new user as a member of this group. This procedure sets up a Linux account that you can use later on when creating and initializing the DBAM database (instead of having to use the **root** account).

You use the **groupadd** command to create the new group. For example, the command:

```
groupadd dirxadm
```

creates the group **dirxadm**. You can use the **useradd** command to create the new user. For example, the command

```
useradd -d /home/spdirx -g dirxadm -c “DirX admin” -s /bin/bash
spdirx
```

creates the user **spdirx** whose home directory is **/home/spdirx**.

You then use Linux commands to create the DirX admin's home directory, give ownership of the directory to this user, and add this user as a member of the DirX Directory administration group. For example:

```
mkdir -p /home/spdirx
chown spdirx /home/spdirx
chgrp dirxadm /home/spdirx
```

The last step is to unlock the new user "DirX admin" by establishing a password for the account. For example:

```
passwd spdirx
New password: xxxxxx
Re-enter new password: xxxxxx
passwd (SYSTEM): passwd successfully changed for spdirx
```

Now you have established a user account from which you can run the DirX Directory commands for configuring, initializing, and loading the database.

#### 3.3.1.2. Assigning Symbolic Links

The next step is to use the assign symbolic links to the raw devices to be used as DBAM data and transaction devices. We carry out this step because it is easier to refer to the devices in subsequent administration commands with symbolic names than it is to use their logical device names.

On Linux, you can use the **fdisk(8)** command to determine the logical device names for the devices. When you use **fdisk -l**, it displays a numbered list of available disks.

Next, you use the **ln** command to create symbolic links to the raw data slices.

On Linux, for example after having created the partitions hda8 and hda9 with the **fdisk** command perform the following commands:

```
ln -s /dev/hda8 /home/spdirx/DBAMdev/DbamTrans1
ln -s /dev/hda9 /home/spdirx/DBAMdev/DbamData1
```

#### 3.3.1.3. Changing Permissions

By default, only user **root** can access raw devices on Linux. The next step is to use the **chgrp** and **chmod** commands to change the permissions on the raw devices so that the members of the DirX Directory administration group can access them.

On Linux, perform the following commands:

```
chgrp dirxadm /dev/hda8
chgrp dirxadm /dev/hda9
chmod g{plus}w /dev/hda8
chmod g{plus}w /dev/hda9
```

When you must install system updates or system patches on your machine you must check the permissions of your raw devices after having installed them. In the event that the installation has changed the permissions you must reset them to the correct values.

## 3.3.2. Installing DirX Directory

The next step is to install the DirX Directory software from the product CD.

You should ensure that there is enough space to accommodate any files you create during your use of the directory service; for example, log files, LDIF content files, and LDIF change files. The amount of space you'll need depends on how you plan to use the example set-up.

### 3.3.2.1. Installing DirX Directory on Linux

Use the **dirxinst** and **dirxinst_root** command. You must be logged in as user in order to run **dirxinst**. To run **dirxinst_root** you must be logged in as root. Follow the installation instructions for Linux given in the *Release Notes*, which you will find using the path **Documentation** > **DirX** → **Readme.txt**. We recommend that you install DirX Directory into a dedicated user account. In our example, this is **spdirx**, as shown in the following example command lines:

- Login as user id:

```
mkdir /usr/tmp/dirx
cd /usr/tmp/dirx
tar xvf <CD-mount-point>/DirX/DirXServer/Linux/dirx*.tar
ksh ./dirxinst /usr/tmp/dirx /home/spdirx
```

- Log in as root:

```
cd /usr/tmp/dirx
ksh ./dirxinst_root /usr/tmp/dirx /home/spdirx
```

## 3.3.3. Creating a Profile for the DBAM Database

Now that we have created our volumes (DBAM raw devices) and installed the DirX Directory software, we next use the DirX Directory administration tool **dbamconfig** to

create a profile for the DBAM database. The profile links the volumes we created earlier to the DBAM data device/transaction device format. To create the profile:

1. Log in as user “DirX admin” (**spdirx**) and run the following **dbamconfig** command:

   ```
   $DIRX_INST_PATH/bin/dbamconfig -c
                        -D “/home/spdirx/DBAMdev/DbamData1”
                        -T “/home/spdirx/DBAMdev/DbamTrans1”
                        -P myProfile
   ```

   We use double quotation marks to ensure that **dbamconfig** (and the other DirX Directory administrative commands used in this example) will be able to parse any space characters that may occur in pathnames (although there are none in this example).

   The command should return the status:

   ```
   Creation of profile ‘myProfile’ was successful
   ```

2. Use the following **dbamconfig** command to display the profile you just created:

   ```
   $DIRX_INST_PATH/bin/dbamconfig -sl -P myProfile
   ```

In the example scenario the output is as follows:

+

```
ProfileName: myProfile
DataDevice(s):
Mountpoint: /home/spdirx/DBAMdev/DbamData1
VolumeName: /dev/hda8
DeviceSize: 136.398 GB
Logical device(s):
BlockType(s): GENERAL | BITSTR | PSEUDO | TREE
DeviceSize:   27.273 GB
BlockType(s): REAL
DeviceSize:   54.546 GB
BlockType(s): AVIDX
DeviceSize:   54.546 GB
TransactionDevice:
Mountpoint: /home/spdirx/DBAMdev/DbamTrans1
```

```
VolumeName: /dev/hda8
DeviceSize: 136.398 GB
```

When we used the **dbamconfig** command to create the profile, we did not specify how the space on the directory data device was to be allocated to the different data types. As a result, **dbamconfig** used a default allocation: 40% of the device to real directory data, 40% to attribute index data, and the remainder to the other types of data.

### 3.3.4. Initializing the DBAM Database

The next step is to use the **dbamboot** command to initialize an empty database that corresponds to the database profile **myProfile** that we just created with **dbamconfig**. We perform this step as user “DirX admin” (**spdirx**).

Run the following **dbamboot** command

```
$DIRX_INST_PATH/bin/dbamboot -P myProfile
```

The command should return the message:

```
DB successfully initialized
DB-limits:
57188608 objects
20 attribute index type(s)
```

Here you can see that the database you’ve initialized can accommodate approximately 57 million entries and 20 attribute index types.

You can use the **dbamdevinfo** command to view the empty database:

```
$DIRX_INST_PATH/bin/dbamdevinfo
DSA profile:            profile1, ID=1
Real object block size: 1 KB
AVIDX cluster size:     32 MB

Logical device:  GENERAL | BITSTR | PSEUDO | TREE
    In use:         0.00 %  (  0.130 MB of   27.270 GB)
    Fragmentation:  0.00 %
Logical device:  REAL
    In use:         0.00 %  (  0.000 MB of   54.539 GB)
    Fragmentation:  0.00 %
```

```
Logical device:  AVIDX
    In use:         0.00 %  (  0.000 MB of   54.546 GB)
    Fragmentation:  0.00 %

Attribute index specific device info:

                Number of indices: 0       (maximum 80)
                Cluster usage:     0.00 %  (  0 of  95)
```

Here you can see how the default allocation for the different directory types has been applied to raw devices for the example scenario on Linux.

### 3.3.5. Loading the Directory Data

Now we are ready to load directory data into the empty database. In this example, we use the **dirxload** administration tool to populate the empty database with sample directory data in an LDIF content file named **Complete_DB.ldif**, which is supplied with the DirX Directory installation in *install_path**/scripts/stand_alone/default*. We perform this task as user “DirX admin” (**spdirx**):

```
$DIRX_INST_PATH/bin/dirxload
         -f
$DIRX_INST_PATH/scripts/stand_alone/default/Complete_DB.ldif
```

The command displays the output:

```
Attribute objectClass(0) INITIAL indexed, file /tmp/0-INITIAL-
19058.iaf
Attribute cn(3) INITIAL indexed, file /tmp/3-INITIAL-19058.iaf
Attribute cn(3) FINAL indexed, file /tmp/3-FINAL-19058.iaf
Attribute sn(4) INITIAL indexed, file /tmp/4-INITIAL-19058.iaf
Attribute sn(4) FINAL indexed, file /tmp/4-FINAL-19058.iaf
Attribute c(6) INITIAL indexed, file /tmp/6-INITIAL-19058.iaf
Attribute c(6) FINAL indexed, file /tmp/6-FINAL-19058.iaf
Attribute o(13) INITIAL indexed, file /tmp/13-INITIAL-19058.iaf
Attribute o(13) FINAL indexed, file /tmp/13-FINAL-19058.iaf
Attribute collectiveOrganizationName(14) INITIAL indexed, file
/tmp/14-INITIAL-19058.iaf
Attribute collectiveOrganizationName(14) FINAL indexed, file /tmp/14-
FINAL-19058.iaf
Attribute ou(15) INITIAL indexed, file /tmp/15-INITIAL-19058.iaf
Attribute ou(15) FINAL indexed, file /tmp/15-FINAL-19058.iaf
```

```
Attribute collectiveOrganizationalUnitName(16) INITIAL indexed, file
/tmp/16-INITIAL-19058.iaf
Attribute collectiveOrganizationalUnitName(16) FINAL indexed, file
/tmp/16-FINAL-19058.iaf
1025 entries created
0 entries rejected
```

Now we have configured and initialized the DBAM database and have loaded directory content into it. We can now start the service and use any LDAP client to view the data. For example, we can use the command-line tool **dirxcp** to view the data in text format, or we can use DirX Directory Manager to view the data in a GUI format.

### 3.3.6. Starting the Service

To start the DirX Directory service, start the **dirxadm** program from the command line, then issue the following **dirxadm** operation:

```
dirxadm> sys start
```

This procedure starts the DirX Directory service. You can now use the following Linux command sequence to check that the service is running:

```
ps -ef | grep dirx
root    24365   469  0 16:00:19 ?  0:00 /home/spdirx/bin/dirxdsas
-d/home/spdirx
spdirx  24366 24365  1 16:00:19 ?  0:00 /home/spdirx/bin/dirxdsa
root    24367 24365  1 16:00:19 ?  0:00 /home/spdirx/bin/dirxldapv3
```

The output shows that the DirX Directory service processes dirxdsa, dirxldapv3 and dirxdsas are all running on the system.

### 3.3.7. Viewing the Directory Data with dirxcp

To use **dirxcp** to view the directory data you loaded into the DBAM database:

1. Start **dirxcp** from the command line.
2. The **dirxcp** program requires that you first bind to the DirX directory service before you can perform any further **dirxcp** operations. Use the following **dirxcp** operation to make an unauthenticated (anonymous) bind to the service:

   ```
   dirxcp> bind -protocol ldapv3
   ```

3. Now you can use **dirxcp** to examine the database. For example, you can use the following **dirxcp** operation to search the database:

```
dirxcp> search o=My-Company -sub -p
1) o=My-Company
2) cn=admin,o=My-Company
3) ou=Sales,o=My-Company
4) ou=Development,o=My-Company
5) ou=Accounting,o=My-Company
6) ou=Product Development,o=My-Company
7) ou=Product Testing,o=My-Company
8) ou=Human Resources,o=My-Company
9) ou=Payroll,o=My-Company
10) cn=Smith John,ou=Sales,o=My-Company
11) cn=Mayer,ou=Sales,o=My-Company
12) cn=Hohner,ou=Sales,o=My-Company
13) cn=Richter,ou=Sales,o=My-Company
14) cn=Abele,ou=Sales,o=My-Company
15) cn=Reichel,ou=Sales,o=My-Company
16) cn=Digger,ou=Development,o=My-Company
17) cn=Filler,ou=Development,o=My-Company
18) cn=Tinker,ou=Development,o=My-Company
:
```

The colon ( : ) at the end of the list indicates that there is more data to display (enter “q[uit]” to discard further output).

# 4. Setting up the DirX Directory Service

Setting up a DirX directory service consists of three main tasks:

1. Installing the DirX Directory software
2. Setting up the DBAM database
3. Setting up the DirX Directory service software

The *Release Notes* describe how to install the DirX Directory software. The chapter "Getting Started with DirX Directory Setup" describes how to set up sample DBAM database configurations, one for Windows and one for Linux. This chapter describes how to set up the DirX Directory service after you have installed DirX Directory and set up the DBAM database.

Setting up the DirX Directory service software consists of the following tasks:

1. Setting up the DSA
2. Setting up the LDAP server
3. Loading directory data into the DSA
4. Starting the service

The chapter describes how to perform these tasks in the context of an example directory service configuration with the following characteristics:

- The disk configuration for the DBAM database is the sample the Windows disk configuration described in the chapter "Getting Started with DirX Directory Setup".
- The DSA is a stand-alone DSA—a DSA that has no superior or subordinate DSAs and manages the entirety of a DIT—with the following characteristics:
  - Its administrative point is at the organization level.
  - It only uses object classes and attribute types defined in the default DSA schema supplied with DirX Directory.
  - It uses simple access control policies for a default administrator and anonymous users.
- The LDAP server has the following characteristics:
  - Its contact DSA is the sample stand-alone DSA.
  - It permits LDAP clients to make updates to the DBAM database.
  - It defines LDAP operation service controls that reflect a non-distributed environment.

The chapter uses a fictitious company called “My-Company” to illustrate the planning decisions and administrative tasks that are involved in setting up a DirX directory service. It discusses the planning considerations and decisions that My-Company must make before it can set up its service, and describes how to build My-Company’s service with the **dirxadm**, **dirxcp**, and **dirxload** programs.

Tcl scripts for building the sample stand-alone DSA and LDAP server can be found in the *install_path***/scripts/stand_alone** directory. These Tcl scripts contain all the commands and procedures discussed in this chapter.

# 4.1. Setting up the DSA

Setting up the DSA consists of the following tasks:

1. Making the planning decisions that are necessary to define the DSA's parameters and administrative framework
2. Bootstrapping the DSA
3. Initializing the DSA

The next sections describe these tasks.

## 4.1.1. Planning the DSA

My-Company is a German company that wants to set up a DSA that contains general information about its employees. All of the information about all of its employees can be stored in a single DSA. The company has no current plans to integrate this DSA into an existing DIT within Germany, but wants to ensure that it can do so in the future if company plans change. Consequently, My-Company chooses to set up its DSA as a **stand-alone DSA** with the appropriate mechanisms in place for future integration with other DSAs.

My-Company needs to make decisions in two areas before its administrators can begin to build their DSA:

- It needs to define the schema
- It needs to define a framework for DSA administration (DSA name, a default administrator, access rights, DUA configuration)

#### 4.1.1.1. Defining the Schema

In order to define the DIT, the administrator must define:

- The attribute types that will be used to store employee information
- The object classes that will be used

The next question that the administrator must consider is:

*Can I use the default DSA schema supplied with DirX Directory, or must I define my own attribute types and object classes?*

The administrator determines that the default DSA schema is sufficient for the company's use, and that he need not extend the DSA schema with custom attribute types and/or object classes. The administrator selects the following object classes from the default DSA schema to represent the company's DIT structure:

- The organization object class to represent the company

- The organizational-unit object class to represent its departments
- The organizational-person object class to represent the employees

The following figure illustrates this structure.

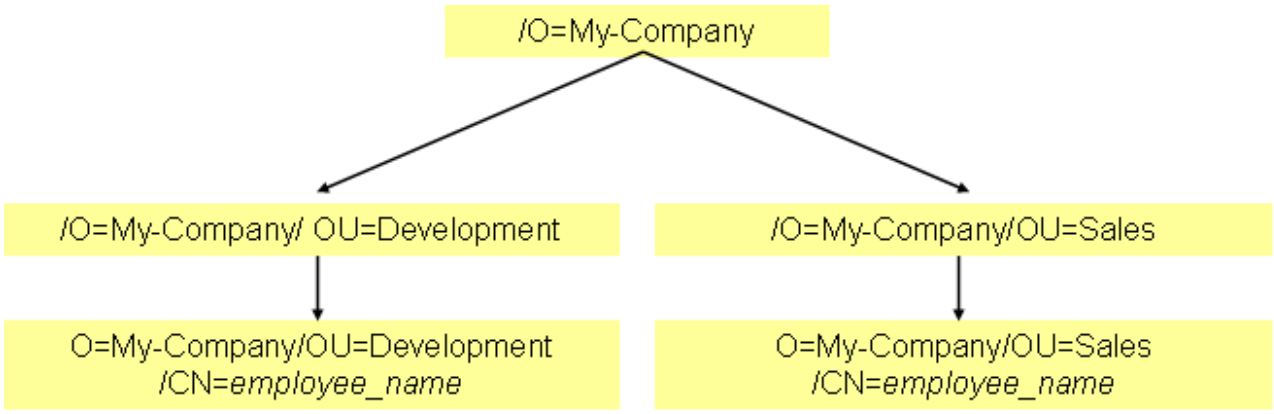



*Figure 23. My-Company Distinguished Name Structure*

My-Company is represented as an object of object class organization (O) and is named by the organization-name **My-Company**, which results in the distinguished name **O=My-Company**.

Sales and Development are represented as objects of object class organizational-unit (OU). The two organizational units use the names **Sales** and **Development** as relative distinguished names (RDNs).

The employees are represented as entries of object class organizational-person (ORP). The entries for the employees are named by the common-name (CN) naming attribute type. Since a large number of employees are located within each organizational unit, My-Company must ensure that each employee is assigned a unique common-name within the organizational unit.

My-Company wants to store the following information in the DSA:

- For the company as a whole, its name, a short description, a telephone number and the postal address
- For either organizational unit, its name
- For each employee, the name of the employee, a short description, the telephone number, an e-mail address and a user password

The standard object classes are associated with a set of standard attribute types that are allowed for the object class. These attribute types are sufficient to store all the information required by My-Company. The following table shows the attribute types of each object class that are used to store the required information in My-Company's DSA.

*Table 1. My-Company Object Classes and Attribute Types*

| Object Class | Stored Information | Attribute Types | Stored Information |
|---|---|---|---|
| organization | company data | organization-name | company name |
| | | description | company description |
| | | telephone-number | company telephone no |
| | | postal-address | company address |
| organizational-unit | organizational unit data | organizational-unit-name | organizational unit name |
| organizational-person | employee data | common-name | employee name |
| | | description | employee description |
| | | telephone-number | employee telephone no |
| | | user-password | employee password |
| | | e-mail | e-mail address |

#### 4.1.1.2. Defining the Administrative Framework

The next planning step for the administrator is to establish the administrative framework in which DSA administration will be performed. This planning process consists of the following tasks:

1. Defining a DSA name and presentation address
2. Defining the location of the first administrative point
3. Defining the default administrator
4. Defining access rights

##### 4.1.1.2.1. *Defining a DSA Name and Address*

A DSA must be given a name and a presentation address in order for it to bootstrap successfully. Establishing a DSA's presentation address is a requirement for network communication between DUA clients and DSAs. Correct administration of the DSA's name and the presentation address is also a requirement for DSA-to-DSA communication if future integration with a superior DSA is planned. Incorrect administration of the DSA's name for example will result in a DIRX_DSA_NAME conflict and the DSA will not be able to run.

The environment variables **DIRX_DSA_NAME** and **DIRX_OWN_PSAP** specify a DSA's name and presentation address. A stand-alone DSA can work with the default values for these environment variables, which are:

DIRX_DSA_NAME="CN=DirX-DSA-*hostname*"

DIRX_OWN_PSAP="TS=DSA1,NA='TCP/IP_IDM!internet=*ip_address*+port=21200'"

where *hostname* is the hostname of the own DSA and *ip_address* is the first IP address for IDM.

#### 4.1.1.2.2. *Defining the Location of the First Administrative Point*

An administrative point establishes the root of an administrative area, which corresponds to the portion of the DIT owned by an organization. Consequently, for My-Company, defining the first administrative point defines the root of My-Company's administrative area, which corresponds to the portion of the DIT that it owns and manages. In order for My-Company to define the location of its first administrative point, it must answer the following question:

*Where does the part of the DIT held by my DSA start?*

The topmost entry held by the DSA must be defined to be an autonomous administrative point, which defines the administrative area as an autonomous administrative area. Defining the area as autonomous means that administration policies of superior DSAs will not affect this DSA.

The first administrative point must also be defined to be access control-specific, which means that the necessary information concerning the access control policies within the company are defined at this level. If no access control policy is defined, then nothing is allowed.

A related question that My-Company must also answer is:

*What names are superior to my DSA?*

The topmost entry held by the My-Company DSA is **O=My-Company**. This is also its context prefix, which is the sequence of Relative Distinguished Names (RDNs) that lead from the root of the DIT to the starting point of a naming context. (A context prefix corresponds to the distinguished name of the starting point of a naming context). My-Company's DIT starts at the organization level directly underneath the root DSE (/). Since it does not plan to integrate with DSAs at the country level or the DC naming scheme, it has no superior names.

#### 4.1.1.2.3. *Defining the Default Administrator*

The process of setting up a DSA requires that a default administrator entry be created. To define a default administrator, the administrator must answer the following question:

*What is the distinguished name for the default administrator, and what other attributes are required for the default administrator?*

The administrator determines that the distinguished name of the default administrator entry is **/O=My-Company/CN=admin**. Since the administrator has planned to have extensive access rights (see the next section), he is expected to use simple authentication. As a result, the administrator must also define a password.

#### 4.1.1.2.4. *Defining Access Rights*

The final planning step for the administrator is to define the access control policy to be applied to users of its DIT. The decisions made in this step determine the contents of the access control-specific subentry that is created as part of DSA initialization. The administrator defines two categories of user:

- The administrator, who is responsible for the DSA administration and therefore needs wide access rights
- All other users, who are allowed to use the DSA to read the information contained in the entries, but are not allowed to modify anything in the DSA.

The administrator will build the DSA and thus needs to be able to create all the information necessary to prepare the DSA for population. Consequently, the administrator must be allowed to

- Modify all user and operational attributes of all entries including the administrative point entry
- Create and modify schema and access control subentries with all their attributes

The administrator is also the only user who is allowed to change the DIT. Therefore, he must be allowed to

- Create all entries with the necessary user and operational attributes
- Delete all entries
- Modify all user and operational attributes of all entries
- Read all user and operational attributes of all entries

When the administrator binds to the DSA, he must identify himself with a name and a password (simple authentication). He is not allowed to read passwords that belong to the employees.

All other users can bind without a name and password (anonymous binding). They are allowed to read all user attributes except for user passwords. They cannot modify anything in the tree.

### 4.1.2. Bootstrapping the DSA

Bootstrapping the DSA means performing the steps that are necessary in order to make the DSA able to come up and understand the first incoming requests using the DAP protocol. These steps are:

1. Creating the first administrative point
2. Creating the default administrator entry

You must use **dirxadm** commands to perform these tasks. Because the **dirxadm** program requires that you perform an administrator bind to the DSA before you can use any other **dirxadm** commands, the first step in the bootstrapping process is to bind to the DSA with **dirxadm**.

#### 4.1.2.1. Binding Anonymously to the DSA with dirxadm

Recall that the My-Company default administrator is to use simple authentication when binding to the DSA. However, since the DSA has not yet been bootstrapped and initialized, the DSA database does not yet exist, and so there is no authentication information available in the DSA. Consequently, the My-Company administrator needs to perform an unauthenticated (anonymous) bind to the DSA in order to use **dirxadm** to bootstrap it. To perform an anonymous bind to the default (and in this case the local) DSA, use the **dirxadm** command:

[**dse**] **bind**

The **dirxadm** object that is used to bind to the DSA is **dse**. Since **dse** is the default object for **dirxcp**, you can omit it in **dirxadm** commands that operate on **dse**. So, for My-Company, the command is:

**bind**

This command performs an anonymous bind to the DSA. An anonymous bind should *only* be used with **dirxadm** during the bootstrap process. The final step in the bootstrapping process should be to control who can use **dirxadm**. This procedure is described in the section “Controlling Access to dirxadm”.

#### 4.1.2.2. Creating the First Administrative Point

The following questions must be answered before creating the first administrative point:

- *Is the standard DSA schema sufficient or is it necessary to create customized attribute types and object classes?*

For My-Company, the standard DSA schema is sufficient.

- *Where does the part of the DIT held by this DSA start, and which superior names must be known?*

For My-Company, the DIT begins at the organizational level below the root DSE (/). The topmost entry held by the DSA is **O=My-Company**.

- *Which access control scheme should be used: basic or simplified? What is the access control policy for the first administrative point? Who is allowed to create, read and modify subentries below the first administrative point?*

For My-Company, the simplified access control scheme is sufficient and the administrator is the only user who is permitted to create, read, modify, and supply subentries to the administrative point.

- *What is the default administrator’s distinguished name and what other attributes are required for the administrator?*

For My-Company, default administrator is named **/O=My-Company/CN=admin** and uses a password.

The administrator can now proceed to create the first administrative point and corresponding administrative entry.

The administrator must use the **dirxadm** program to create the glue and the first administrative point. Once DSA bootstrapping is complete and the appropriate access rights are set, he must create all other entries and subentries with the **dirxcp** program. The **dirxadm** object that is used to create glues and entries is **dse**. Since **dse** is the default object, it may be omitted in the command line.

To create the first administrative point, use the **dirxadm** command:

[**dse**] **create** *distinguished_name* **-attribute** *attribute_list*

Supply the DN of the first administrative point in the *distinguished_name* argument. The attribute list must contain the following attributes:

- The object-class (OCL) user attribute with the names of all object classes of the entry
- The DSET operational attribute with the following values:
- ADM_POINT, which indicates that the entry is an administrative point
- CP, which indicates that the administration point represents a context prefix of a naming context
- ENTRY, which indicates that the administration point is an entry; this value permits the entry to hold user attributes and makes it visible to the DAP protocol
- The access-control-scheme (ACS) operational attribute, which specifies whether basic access control (BACS) or simplified access control (SACS) is to be used
- The administrative-role (AR) operational attribute with the following values:
- Autonomous-area (AA), to define the entry as an administrative point of an autonomous administrative area
- Access-control-specific-area (ACSA), to allow the creation of associated access control-specific subentries
- The subentry-ACI (SACI) operational attribute, which specifies the access rights for the subentries of the first administrative point

If advanced access control policies will be used, additional attributes may be required.

The following **dirxadm** command creates a first administrative point for My-Company:

```
create /O=My-Company -attr \
       {OCL=TOP;ORG} \
       {DSET=ENTRY+ADM_POINT+CP} \
       {AR=AA;ACSA}  \
       {ACS=SACS} \
       {SACI={ID=admin: enable Handling of Subentries,
              PR=254,
              AL={BL={L=SIMPLE}},
```

```
            UF={UC={N={DN={/O=My-Company/CN=admin}}},
                UP={PI={E=TRUE},
                    GAD=grantRead+grantBrowse+\
                        grantDiscloseOnError+\
                        grantReturnDN+\
                        grantAdd+grantRemove+grantModify};
                   {PI={AT=SS;OC;AT;DSR;
                        DCR;MR;MRU;CRN;CRT;MN;MT;PACI;EACI,
                        AAV=SS;OC;AT;DSR;
                        DCR;MR;MRU;CRN;CRT;MN;MT;PACI;EACI,
                        AUATV=TRUE},
                    GAD=grantDiscloseOnError+\
                        grantRead+grantCompare+\
                        grantFilterMatch+\
                        grantAdd+grantRemove} } } } \
"CRN={/O=My-Company/CN=admin}" \
{PA={PA1=My-Company,PA2=ABC Street 123,\
     PA3=D-01234 City,PA4=Germany}} \
{TN=+49 12 345 67 890} \
{DSC=My-Company }
```

This command adds an administrative entry to the DIT with the distinguished name **/O=My-Company** and with the structural object class (OCL) organization (ORG). The entry specifies a set of operational attributes:

- The DSET attribute defines the object to be an ordinary entry (ENTRY), an administrative point (ADM_POINT) and a context prefix (CP).
- The AR of the administrative point is Autonomous Area (AA), Access Control Specific Area (ACSA).
- The ACS attribute specifies that the simplified access control (SACS) shall be used

The SACI attribute is a multivalued structured attribute that defines the access rights to create and modify subentries of the first administrative point.

Each value of an SACI attribute must be assigned an Identification-Tag (ID) which is used to distinguish between attribute values. The tag should give a short description of the purpose of the value. For My-Company, the ID indicates that the value applies to the default administrator, who is permitted to handle subentries.

The Precedence (PR) of the value is relevant if more than one SACI attribute value applies to an entry. The values with higher precedence overrule values with lower precedence. The range of integer values used is 0 to 255. For My-Company, the value of PR=254 indicates nearly the highest possible precedence.

The Authentication-Level (AL) defines the minimum required level of authentication that

the user must have undergone before this SACI attribute value is considered for granting access rights. The value of the Subentry-ACI attribute applies only to users who use at a minimum the specified authentication level. My-Company expects at a minimum simple authentication.

In the User-First (UF) component, access rights for individual users or groups of users are granted or denied. The example specifies in the User-Class (UC) component that the permissions are made for the default administrator.

The User-Permissions (UP) component defines two values as Protected-Item (PI). The parameter (E=TRUE) itself defines Grants-And-Denials (GAD) to read, browse (list and search), add, remove and modify the subentries of the first administrative point. Furthermore, all operational attribute types (AT) and values (AAV) which can occur in a subentry and all collective attribute types and values (AUATV) may be read and modified and are allowed to be used in search filters. All grants are valid for the specified user, the default administrator, and concern subentries of the first administrative point only. The grantDiscloseOnError ensures that a specific error message (insufficient access rights) is returned if the access rights are not sufficient for an operation. Otherwise, a general error message is returned.

The creators-name (CRN) attribute specifies the name of the default administrator as creator. This attribute is not required for correct directory operation but should be supplied for the consistency and completeness of the data. For My-Company, the default administrator is the only user authorized to add entries after DSA setup is complete. The default administrator must use **dirxcp** to add entries; **dirxcp** automatically names the administrator as creator of the entry. Therefore, the default administrator should also appear as the creator of the first entry that is created with **dirxadm**.

A set of user attributes is also specified for the first administrative point:

- The Postal-Address (PA) and Telephone-Number (TN) attributes specify the address and telephone number of the My-Company
- The description (DSC) attribute is used to give a more detailed description of the My-Company

#### 4.1.2.3. Creating the Default Administrator Entry

To create the entry for the default administrator, use the **dirxadm** command:

[**dse**] **create** *distinguished_name* **-attribute** *attribute_list*

Supply the distinguished name of the default administrator determined during the administration framework planning process in the *distinguished_name* argument (see “Defining the Default Administrator”).

The attribute list must contain at a minimum the following attributes:

- The object-class (OCL) user attribute with the names of all object classes of the administrator entry.
- All mandatory non-naming user attributes of the object class of the administrator entry.

- A user password should be supplied if the access control policy established during the administrative framework planning process requires it (see “Defining Access Rights”).

Additional attributes to the ones listed here can be provided; for example, an attribute can be supplied to give a description of the administrator.

For My-Company, an administrator with the distinguished name **/O=My-Company/CN=admin** must be created. The entry is of object class organizational-person, and a user password shall be used during authentication. The following **dirxadm** command creates the default administrator entry for My-Company:

```
create  /O=My-Company/CN=admin \
    -attr   {OCL=TOP;PER;ORP} \
            {DSET=ENTRY} \
            {SN=admin} \
            {DSC=Default Administrator} \
            {UP=dirx} \
            {CRN={/O=My-Company/CN=admin}}
```

The command adds the default administrator entry to the DIT with the distinguished name **/O=My-Company/CN=admin** and with the object classes (OCL) Person and Organizational-Person (ORP). (Since the organizational-person object class is derived from person, both must be named.) The object class requires that a surname (SN) be specified, for which the same value as for the common-name is used. As additional information, the description (DSC) attribute is specified to indicate that the entry specifies the default administrator. The user-password (UP) of the administrator is set to the value **dirx**. For consistency and completeness, the creators-name (CRN) attribute is also specified.

#### 4.1.2.4. Controlling Access to dirxadm

During the DSA bootstrapping process, unauthenticated (anonymous) binds are permissible to permit the administrator to use **dirxadm** while there is not yet any authentication information in the DSA database. However, once a DSA has been bootstrapped, the use of **dirxadm** should be restricted to those users who are experienced DirX Directory administrators.

Consequently, the final step in the bootstrapping process is to enable access control for using **dirxadm**. The DirX Administrators (DADM) attribute in the root DSE specifies who can use **dirxadm** to bind to a DSA. The default administrator must add the distinguished name of his entry to the DirX Administrators attribute. To add an administrator entry to the DirX Administrators attribute, use the **dirxadm** command:

**modify / -addattr DADM=***administrator-entry-distinguished-name*

For My-Company, the default administrator issues the **dirxadm** command:

```
modify / -addattr DADM={/O=My-Company/CN=admin}
```

The administrator can now use **dirxadm** to bind to the DSA with simple authentication and the password **dirx**; no one else is permitted to bind with **dirxadm**.

### 4.1.3. Initializing the DSA

The next step in building the DSA is to create subentries of the first administrative point. Subentries hold operational information that is valid for a specific subset of the entries in an administrative area. For this stand-alone DSA, the following subentry must be created:

- An access-control-specific subentry, which defines the access control policy to be applied to the DIT

You use the **dirxcp** program to create these subentries. Because **dirxcp** is a DUA and uses DAP protocol to perform its operations on a DSA, you must first:

- Define a DUA presentation address for **dirxcp**
- Make the DUA presentation address and the DSA name and presentation address available to **dirxcp**
- Bind with **dirxcp** to the DSA with the appropriate authentication

The section “Defining a DUA Presentation Address” describes how to establish a presentation address for a DUA. The next sections describe how to establish the DUA and DSA addresses in the client, how to bind to the DSA, and how to create the access-control specific subentry.

#### 4.1.3.1. Establishing the DUA and DSA Addresses in the Client

The name of the DSA and its presentation address must be made available as the default DSA to the **dirxcp** program. The DUA presentation address must also be established.

The DUA presentation address and DSA name and presentation address are provided to the client in the file **dirxcl.cfg**, which is located in the *install_path*/**client/conf** subdirectory. The first non-commented line of the file specifies the DUA presentation address. Next, a list of DSA names and addresses is specified. The first DSA name in the list is the default DSA. To make the DSA available to **dirxcp**, you must modify this line to contain the DSA name and address.

In the case of My-Company, their administrator modifies the client configuration file to contain the name and presentation address of the DUA and the name and presentation address of DirX-DSA as the default DSA:

```
# File: dirxcl.cfg
# first non-comment line is the client address

Self TS=Client1,NA='TCP/IP_IDM!internet=192.168.19.129+port=2222';
'TCP/IP!internet=192.168.0.12+port=2222'
# first line is the default DSA
# the next two lines must be on one line in the file
```

```
/CN=DirX-DSA-MyHostname
TS=DSA1,NA='TCP/IP_IDM!internet=192.168.19.129+port=21200'
```

#### 4.1.3.2. Binding to the DSA with dirxcp

The **dirxcp** program requires that you first bind to a DSA before you can use any other **dirxcp** commands. Recall that the My-Company administrator is to use simple authentication for binding to the DSA. To bind to the default (and in this case the local) DSA using simple authentication, use the **dirxcp** command:

[**obj**] **bind -authentication** *auth_method* **-user** *username*
*_*-password* _password*

The **dirxcp** object that is used to bind to the DSA is **obj**. Since **obj** is the default object for **dirxcp**, you can omit it in **dirxcp** commands that operate on **obj**.

To bind to the bootstrapped DSA with simple authentication, the My-Company administrator uses the command:

```
bind -authentication SIMPLE -user /O=My-Company/CN=admin
     -password dirx
```

#### 4.1.3.3. Creating the Access Control-Specific Subentry

The access control-specific subentry defines the access control policy for a subset of the administrative area with which it is associated. An access control specific subentry of the first administrative point must define the access control policy applicable in the DSA. By default, no user is granted any access right to any entry of the DIT.

To create an access control-specific subentry below the first administrative point, use the **dirxcp** command:

[**obj**] **create** *distinguished_name* **-attribute** *attribute_list*

Supply the name of the access control-specific subentry in the *distinguished_name* argument; in the case of My-Company, it is **/O=My-Company/CN=AccessControl-Subentry**. Note that you can create an access control-specific subentry only if the administrative-role defined in the administrative point is Access Control Specific Area (ACSA).

The attribute list must contain at a minimum the following attributes:

- The object-class (OCL) user attribute, which specifies the Subentry (SUBE) and access-control-subentry (ACS) object classes.
- The subtreeSpecification (SS) operational attribute, which should specify its default value, indicating that the subentry holds for the whole subtree.
- The prescriptive-ACI (PACI) operational attribute, which lists the applicable access

control items.

For My-Company, the access policy is simple: the administrator is allowed to execute operations on all entries except for reading user passwords. All other users are allowed to read and search the user attributes of ordinary entries. The following **dirxcp** command creates the access control-specific subentry for My-Company below the first administrative point:

```
    create /O=My-Company/CN=AccessControl-Subentry  -attr \
           {OCL=SUBE;ACS} \
           {SS={DEF=TRUE}} \
           {PACI={ID=admin: enable most of operations \
                    but no Rename,
                  PR=254,
                  AL={BL={L=SIMPLE}},
                  UF={UC={N={DN={/O=My-Company/CN=admin}}},
                      UP={PI={E=TRUE},
                      GAD=grantDiscloseOnError+\
                          grantRead+grantBrowse+\
                          grantReturnDN+\
                          grantAdd+grantRemove+grantModify};
                      {PI={AT=AR;ACS;SACI;CRN;
                           CRT;MN;MT;SOC;GSR;CE;EACI,
                           AAV=AR;ACS;SACI;CRN;
                          CRT;MN;MT;SOC;GSR;CE;EACI,
                          AUATV=TRUE},
                     GAD=grantDiscloseOnError+\
                         grantRead+grantCompare+\
                         grantFilterMatch+\
                         grantAdd+grantRemove}

} };
                  {ID=Public Access: enable Read and Search - \
                      disable read password,
                   PR=0,
                   AL={BL={L=NONE}},
                   UF={UC={AU=TRUE},
                       UP={PI={E=TRUE},
                           GAD=grantDiscloseOnError+\
                               grantRead+\
                               grantBrowse+grantReturnDN};
                          {PI={AUATV=TRUE},
```

```
                          GAD=grantDiscloseOnError+\
                              grantRead+grantCompare+\
                              grantFilterMatch };
                         {PR=255,
                          PI={AT=UP},
                          GAD=denyRead} } } }
```

The prescriptive-ACI (PACI) attribute of the access control subentry specifies the access rights of the different users to the entries of the specified subtree. Two values of the attribute specify the access rights of the administrator and of anonymous users.

Each value of a PACI attribute is assigned an Identification-Tag (ID) that gives a short mnemonic reference to the value. The first ID indicates that the value is applicable to the administrator, who is enabled to execute most operations except for reading a user-password. The precedence (PR) and the authentication-level (AL) of the first PACI attribute value are handled as for the subentry-ACI attribute of the first administrative point.

In the user-first (UF) component, individual users or groups of users are granted access rights.The first user-first (UF) element specifies in the user-class (UC) that the permissions are made for the default administrator.The user-permissions (UP) define two values as Protected-Item (PI).As the first item, the entries (E) get grants-and-denials (GAD) to read, browse, add, delete and modify the entry.As the second item, all operational attribute types (AT) and values (AAV) of an entry and all user attribute types and values (AUATV) can be read, added or deleted.All these grants are valid for the specified user, the administrator, and concern all subordinate entries of the first administrative and the administrative point itself.

The second ID indicates that the value is applicable to anonymous users, who are enabled to read and browse entries.The precedence (PR) is set to 0 so that this access control is overruled by any other rule.The authentication-level (AL) of the second PACI attribute value specifies anonymous bind as the minimum level of authentication.

The second PACI user-first (UF) component specifies in the user-class (UC) that the permissions are made for all users (AU).The user-permissions (UP) define three groups of protected-items (PI).First, the entries may be read and browsed.Secondly, all user attribute types and values (AUATV) may be read and used in search filters.The third item protects the user-password from being read.

With the example setting, the reading of user passwords is denied to all users with the highest precedence (255).

Now the DSA is set up and the My-Company administrators can proceed to set up the LDAP server.

## 4.2. Setting up the LDAP Server

Setting up the LDAP server consists of the following steps:

- Establishing the contact DSA for the LDAP server
- Creating the LDAP server subentries in the contact DSA
- Modifying the subentry ACI of the administration point on the contact DSA to permit the LDAP server to read the LDAP server subentries

### 4.2.1. Planning the LDAP Server

My-Company's stand-alone DSA contains general information about its employees.This DSA is to be located at the company's headquarters in Munich.My-Company also employs a travel agency in Frankfurt to handle its employee business travel arrangements.The travel agency needs to have access to the My-Company employee information, and the employees at the travel agency have access to the Internet and use a variety of browsers that contain built-in LDAP clients.The My-Company administrators need to set up the DirX Directory LDAP server to make LDAP access to the My-Company employee database available to the travel agency in Frankfurt.The administrators also want access to the DSA via the LDAP server so that they can use DirX Directory Manager for simple maintenance tasks.The following figure illustrates this scenario.

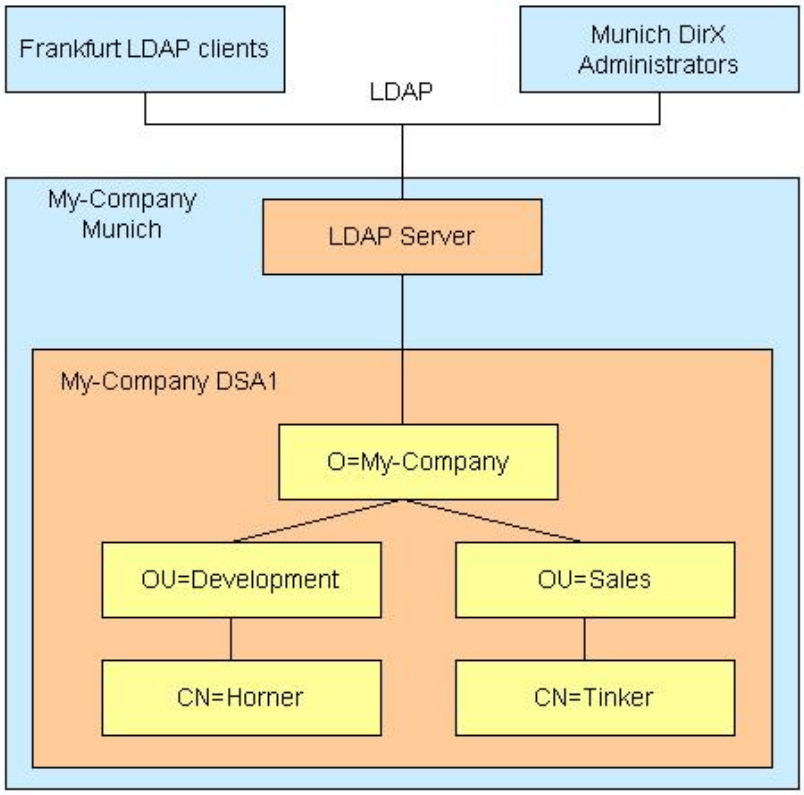



*Figure 24. My-Company LDAP Configuration*

My-Company needs to make decisions in the following areas before its administrators can begin to build the LDAP server configuration:

- It must define the capabilities of its LDAP server

- It must define the framework for LDAP server administration (contact DSA, LDAP server port number, LDAP server administrator)
- It must define the environment in which its LDAP server operates

#### 4.2.1.1. Defining the LDAP Server Capabilities

My-Company must answer the following questions about its LDAP server capabilities:

- *Will old LDAP v2-only clients need to communicate with the LDAP server, and if so, how is the LDAP server to handle LDAP v2 requests?*

  The old LDAP v2 protocol has a number of limitations in comparison to LDAP v3 protocol. For example, it does not support X.509 (v3) certificates, and it supports only the printable-string character set encoding for attribute values that are strings. It also does not support sub-typing (for example, lang-de) and requires special mechanisms to handle binary values (for example no support of ;binary). However, many LDAP v2-only clients exist, and these clients may need access to the directory information made available through a DirX Directory LDAP server that supports both the LDAP v2 and v3 protocol.

  If an LDAP server is to support LDAP v2 operations, it needs to know:

  - The character set encoding to use when converting printable-string attribute values supplied in LDAP v2 requests. This decision affects the value supplied in the LCCQ (ldapCharsetConvRequest) attribute of the LDAP configuration subentry; see the section "Creating the LDAP Configuration Subentry" for further details.
  - The character set encoding to use for search results generated by LDAP v2 operations This decision affects the value supplied in the LCCS (ldapCharsetConvResult) attribute of the LDAP configuration subentry; see the section "Creating the LDAP Configuration Subentry" for further details.
  - The representation to use when returning attribute values that are X.509 certificates to LDAP v2 clients This decision affects the value supplied in the LAHE (ldapASN1Header) attribute of the LDAP configuration subentry; see the section "Creating the LDAP Configuration Subentry" for further details.

  My-Company determines that the Frankfurt travel agency has a number of LDAP v2-only clients that need to communicate with the My-Company LDAP server. It decides to use the universal-string **UTF-8 character set encoding** for v2 requests that contain directory-string attribute values and for encoding the search results of v2 operations. It decides to return X.509 certificates to LDAP v2 clients in **binary ASN.1** format.

- *Is the LDAP server to allow updates to the DSA by LDAP clients?*

  If users who are accessing a directory through an LDAP server are to be prohibited from making updates to that directory, setting up the LDAP server as a read-only server will improve the performance of the directory service. However, in the case of My-Company, the My-Company administrators want to be able to use DirX Directory Manager to make administrative changes to My-Company's DIT. It is also possible that the Frankfurt travel agency will need to modify My-Company's DIT; for example, to include airline seating preferences and other employee-specific travel information. Consequently, My-

Company decides to allow the LDAP server to permit updates to the My-Company DSA by travel agent LDAP clients. This decision affects the value supplied in the LROS attribute of the LDAP configuration subentry; see the section "Creating the LDAP Configuration Subentry" for further details.

Note, however, that regardless of an LDAP server's status (read-only or read-write), the contact DSA performs access control verification using the selected access control scheme on all modification requests from LDAP clients.

- *What LDAP connection parameters should be established for the LDAP server?*

  LDAP connection parameters define the LDAP operating environment for the LDAP server. LDAP connection parameters include:

  - The maximum number of concurrent LDAP connections that the LDAP server can support; this value is specified in the LMCO (ldapMaxConnections) attribute of the LDAP configuration subentry. This parameter is closely related to the number of file descriptors on a system and should be adjusted accordingly. Note that each LDAP connection may require two socket descriptors—one for the LDAP client connection and one for the DAP connection to the DSA—if backend sharing is not possible.
  - The maximum amount of time that an idle LDAP connection can remain open before the server closes it; this value is specified in the LCIT (ldapConnectionIdleTime) attribute of the LDAP configuration subentry.
  - Whether the LDAP server closes authenticated DAP connections immediately when no more references are present; the LUDT (ldapUnbindDelayTime) attribute of the LDAP configuration subentry enables and disables this feature. Closing DAP connections immediately can save system resources (memory and socket descriptors) at the cost of performance.
  - Whether the LDAP server will use the same DAP connection for the same authenticated users to perform DSA operations; the LBS (ldapBackendSharing) attribute of the LDAP configuration subentry enables and disables this feature. If backend sharing is allowed, users with identical credentials will use the same backend connection. The value of this attribute affects the amount of system resources used.
  - Whether the LDAP server allows anonymous users access to the database. The LDAA (ldapDenyAnonymousAccess) attribute enables and disables this feature. Note that prohibiting anonymous access does not conform to RFC standards and may cause LDAP clients to malfunction when reading the LDAP root or the LDAP schema.

The hardware and software features and limitations of the machine (for example, file descriptors, socket connections already in use) on which the LDAP server is to run determine the LDAP connection parameters that need to be set up for an LDAP server. The My-Company administrators need to select LDAP connection parameters that match the capabilities of the machine on which they choose to install and run the LDAP server.

#### 4.2.1.2. Defining the LDAP Server Administration Framework

My-Company must answer the following questions about its LDAP server administration environment:

- *What port will the LDAP server listen on for LDAP requests*?

  An LDAP server needs to be assigned a port on which to listen for incoming requests from LDAP clients. In general, LDAP servers are assigned the port number **389**. In some cases, however, this port may be in use by another service, and so a different LDAP server port must be selected from the available ports. Since the machine on which the LDAP server is to run has nothing assigned to port **389**, My-Company decides to use this port number for its LDAP server. This value will be specified in the LPNU (ldapPortNumber) attribute of the LDAP configuration subentry.

- *What will be the LDAP server's contact DSA?*

  An LDAP server's contact DSA is the DSA that can make available the portion of the DIT which LDAP clients need to access. Through the LDAP server, LDAP clients will have access to the attribute types and object classes in the contact DSA's system schema and will have access to the naming contexts supported by the DSA. The selection of the contact DSA also defines who is allowed to set up the LDAP server configuration, because the LDAP server setup procedure involves administering the contact DSA (for example, creating new LDAP-specific subentries and modifying the subentry ACI)

  In the case of My-Company, the travel agency needs access to the employee information contained in the stand-alone DSA **CN=My-Company-DSA1**. This DSA masters a single naming context, which is **/O=My-Company**. The administrator for this DSA is the default My-Company administrator **/O=My-Company/CN=admin**.

  The file *install_path*/ldap/conf/dirxldap.cfg* contains the setting for the contact DSA to which the LDAP server will try to forward the LDAP requests.

- *How many LDAP servers will the contact DSA support?*

  In some cases, it can be useful to set up more than one LDAP server process on the same system. In this configuration, each LDAP server supports different capabilities—for example, to handle a particular character set requirement of a group of LDAP clients—and listens on its own separate port for requests from these clients. The set of LDAP servers, however, shares the same contact DSA. My-Company decides that its initial LDAP configuration is to consist of **one LDAP server**, with the possibility of adding multiple servers later on.

- *Will the LDAP server support SSL/TLS connections?*

  The Secure Socket Layer (SSL) and Transport Layer Security (TLS) protocols allow for mutual authentication between clients and servers based on public key/private key cryptography and permit the transmission of encrypted messages between clients and servers. The SSL and TLS protocols provide clients and servers with a protection mechanism for the exchange of confidential data such as financial and medical information over untrusted TCP/IP networks. The DirX Directory LDAP server can

support both SSL and TLS protocols. To use SSL/TLS, a special configuration subentry must be created in the database, and the clients must be provided with a Public/Private Key Infrastructure (PKI); for example, certificates.

Because My-Company does not expect its LDAP clients to be exchanging sensitive data with its LDAP server, it decides that **its initial LDAP configuration will not support SSL/TLS** connections, with the possibility that it may do so in the future.

- *Should the LDAP server's audit facility be enabled?*

  Companies can sometimes require auditing of all requests that the LDAP server performs. The LDAP server provides auditing of every processed request and important data associated with each request. One audit record contains all the information necessary for billing purposes or for the administrator to determine whether clients are using the service correctly. To enable LDAP server auditing, an extra audit subentry must be created in the database.

  Because My-Company's directory is not managing sensitive data, the administrator decides **not to use the LDAP server audit facility**.

- *Should the LDAP server's result cache be enabled?*

  Enabling the LDAP server's result cache can significantly increase the performance of search operations in cases where many anonymous users or users using the same authentication perform the same search operations.

  Because My-Company expects that each user authenticates with individual credentials and that a rather high percentage of modifications are performed, the administrator decides to **disable the LDAP server's result cache**.

- *How many LDAP clients are working in parallel?*

  The number of clients that are issuing requests in parallel to the LDAP server may require a change to the internal thread pool size. This pool size limits the number of parallel operations in the LDAP server. (Note the difference between parallel connections, which can be idle, and parallel operations, which consume one thread from the pool). If the pool is exhausted, incoming new requests are queued. The LTPS (ldapThreadPoolSize) attribute specifies the maximum number of threads for the LDAP server thread pool.

  Because My-Company owns a fast machine and expects no extraordinary peaks of LDAP requests, the administrator decides to **keep the default thread pool size** unless he experiences problems related to parallel LDAP requests.

### 4.2.2. Establishing the Contact DSA

The LDAP server cannot initialize successfully unless it knows which DSA is available to it. The name and presentation address of this contact DSA must be made available to the LDAP server as the default DSA. When the LDAP server initializes, it automatically performs an anonymous DAP bind to this contact DSA and reads its configuration information, which is stored in LDAP-specific subentries within the DSA.

The contact DSA's name and presentation address are provided to the LDAP server in the file **dirxldap.cfg**, which is located in the *install_path*/**ldap/conf** directory. This file must also contain the LDAP server's DUA presentation address, because the DUA presentation address contains the network type to be used for remote connections.

In the case of My-Company, their administrator modifies the LDAP server configuration file **dirxldap.cfg** to contain the name and presentation address of the stand-alone DSA **My-Company-DSA1** as the contact DSA and the DUA presentation address of the LDAP server.

```
#File: dirxldap.cfg
#the first line is the contact (default) DSA

/CN=DirX-DSA-MyHostname
TS=DSA1,NA='TCP/IP_IDM!internet=192.168.19.129+port=21200'
#the line starting with the keyword self is the LDAP server

Self TS=ldapServer,NA='TCP/IP_IDM!internet=192.168.19.129+port=2222'
```

### 4.2.3. Creating the LDAP Server Subentries

Three basic subentries are associated with a DirX Directory LDAP server:

- The LDAP root subentry
- The LDAP global schema subentry
- The LDAP configuration subentry

The LDAP root subentry is a read-only subentry that the DSA creates during its initialization procedure. The LDAP root subentry provides information about the contact DSA's DIT and the supported features of the LDAP server, for example LDAP v3 control extensions like password policy control. LDAP clients read the LDAP root subentry to obtain the information contained within it. See the section "Attributes of the LDAP Root Subentry" in the *DirX Directory Syntaxes and Attributes* for complete details about the attributes of the LDAP root subentry.

The LDAP global schema subentry is a read-write subentry that the DSA creates during its initialization procedure. The LDAP global schema subentry defines the object classes and attribute types available to LDAP clients in an LDAP server. When it creates the LDAP global schema subentry, the DSA uses its system schema to build a list of available object classes and attribute types and stores these lists as values of the LDAP OC (LOC) and LDAP AT (LAT) attributes of the LDAP global schema subentry. The LDAP server reads the DSA's LDAP global schema subentry to build its LDAP global schema. LDAP clients can read the LDAP global schema in order to retrieve information about attribute types and object classes. LDAP clients can also make updates to the DSA schema by issuing LDAP modify requests to the LDAP global schema; the LDAP server updates the LDAP global schema subentry and the DSA system schema. Note that LDAP clients can only update the schema if they have the appropriate access rights. The contact DSA automatically creates the LDAP root subentry and the LDAP global schema subentry below the root DSE when it is started

for the first time. It creates the LDAP root subentry with the distinguished name **/CN=ldapRoot** and creates the LDAP global schema subentry with the distinguished name **/CN=Schema**.

The LDAP configuration subentry defines:

- The port numbers that the LDAP server is to use (attributes LPNU and LSPN)
- The LDAP connection parameters for the LDAP server (attributes LMCO, LCIT, and LUDT)
- LDAP operation service controls
- Whether the LDAP server is read-only or read-write (attribute LROS)
- How to handle requests from LDAP v2-only clients (attributes LCCQ, LCCS, and LAHE)
- Whether or not the LDAP server enables result caching (LCA, LCAP)
- Whether to use a special thread pool size (LTPS)

There can be more than one LDAP configuration subentry per DSA; this is mandatory if the configuration is to support multiple LDAP server processes. The administrator of the contact DSA must create the LDAP configuration subentries below the first context prefix using **dirxcp** and must give it the common name **ldapConfiguration**.

The following figure illustrates the locations of the LDAP server subentries for My-Company's stand-alone DSA.

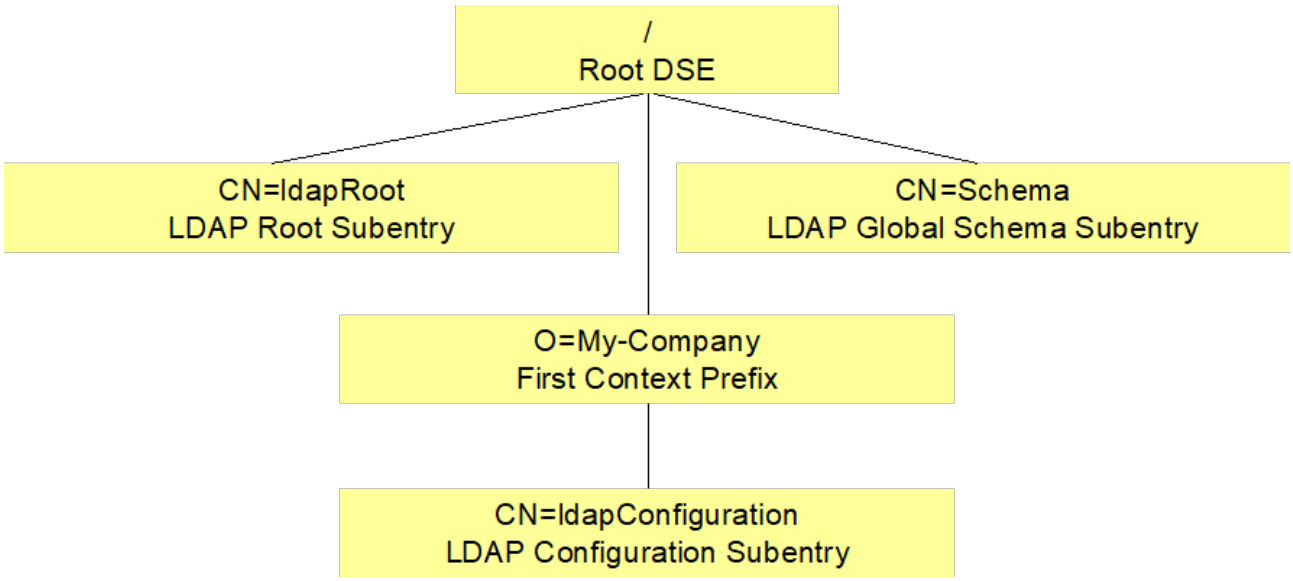



*Figure 25. LDAP Server Subentry Locations*

Creating the LDAP configuration subentry on the contact DSA consists of the following steps:

1. Binding to the contact DSA
2. Creating the LDAP configuration subentry with **dirxcp**

In order to carry out these tasks, the administrator of the contact DSA must have access rights to create and modify subentries on the DSA and must be an authorized DirX

Directory administrator.

In the case of My-Company, the stand-alone DSA setup process has established the My-Company default administrator with access rights to create and modify subentries and has also established him as an authorized DirX Directory administrator for the stand-alone DSA (by virtue of adding his distinguished name to the DADM attribute).

Creating the LDAP configuration subentry with **dirxcp** is most easily performed in two steps:

1. Creating the subentry with attributes for port numbers, connection parameters, LDAP v2 client management, and server type (read-only or read-write)
2. Adding the LDAP operation service control attributes to the subentry

#### 4.2.3.1. *Creating the Subentry*

To create the LDAP configuration subentry below the first context prefix (which in the case of My-Company is also the first administrative point), use the **dirxcp** command:

[**obj**] **create** *distinguished_name* **-attribute** *attribute_list*

Supply the distinguished name of the LDAP configuration subentry in the *distinguished_name* argument; it is **/O=My-Company/CN=ldapConfiguration** in the case of My-Company. The *attribute list* parameter can contain one or more LDAP-specific attributes. These attributes are optional; if an LDAP-specific attribute is not supplied, the DirX Directory LDAP server uses a default value.

For My-Company, the attribute list contains the following LDAP-specific attributes:

- The LDAP port number (LPNU) user attribute, which specifies the RPC port on which the LDAP server is to listen for LDAP client requests. For My-Company, it is **389** (which is the default value).
- The LDAP secure port number (LSPN) user attribute, which specifies the port number on which the LDAP is to listen for Secure Socket Layer (SSL) LDAP protocol. For My-Company, this value is **0** because it has chosen not to enable SSL/TLS connections (the default value).
- The LDAP Max Connections (LMCO) user attribute, which specifies the maximum number of simultaneous LDAP connections that the LDAP server supports. My-Company selects **1024** (the default value is **4000**).
- The LDAP Connection Idle Time (LCIT) user attribute, which specifies the number of seconds of inactivity after which the LDAP server is to close an LDAP connection to an LDAP client. My-Company chooses **300** seconds (which is the default value).
- The LDAP Unbind Delay Time (LUDT) user attribute, which controls whether the LDAP server closes authenticated DAP connections. A value of **0** in this attribute directs the LDAP server to close the connections after the last LDAP client has performed an unbind for this connection; a positive integer value directs the LDAP server to close them that number of seconds after the last LDAP connection using this connection is closed. My-Company chooses to close authenticated DAP connections after the last LDAP client has unbound from this connection (which is the default value).

- The LDAP ASN1 Header (LAHE) user attribute, which specifies whether X.509 certificates in attributes are returned in binary ASN1 (a value of **FALSE**) or hexdump ASN1 format (a value of **TRUE**) in the results of LDAP version 2 operations. My-Company selects to return X.509 attributes in binary ASN1 (which is the default value).
- The LDAP Character Set Conversion Request (LCCQ) user attribute, which specifies the representation of string attributes in LDAP version 2 operations. The possible representations are **LATIN-1**, **UTF8**, and **T61**. My-Company chooses **UTF8** (which is the default value).
- The LDAP Character Set Conversion Result (LCCS) user attribute, which specifies the conversion that the LDAP server is to perform for search results of LDAP version 2 operations. The possible representations are **LATIN-1**, **UTF8**, and **T61**. My-Company chooses **UTF8** (which is the default value).
- The LDAP Read Only Server (LROS) user attribute, which controls whether the LDAP server allows update operations. The possible values are **FALSE** (allow updates) and **TRUE** (prohibit updates). Since My-Company has decided that its LDAP server will permit update operations on the stand-alone DSA, it selects **FALSE** (which is the default value).

The attribute list must also contain at least the following subentry-specific attributes:

- The object class (OCL) user attribute that specifies the Subentry (SUBE) and LDAP configuration (LCFG) object classes.
- The subtree-specification (SS) operational attribute, which must specify its default value, indicating that the subentry holds for the entire subtree.

The following **dirxcp** command creates an LDAP configuration subentry for My-Company:

```
create /O=My-Company/CN=ldapConfiguration -attr \
       {OCL=SUBE;LCFG} \
       {SS={DEF=TRUE}} \
       {LPNU=389} \
       {LSPN=0} \
       {LMCO=1024} \
       {LCIT=300} \
       {LUDT=0} \
       {LAHE=FALSE} \
       {LROS=FALSE} \
       {LCCQ=UTF8} \
       {LCCS=UTF8}
```

#### 4.2.3.2. *Adding LDAP Operation Service Control Attributes*

To add the attributes that define the service controls to be applied to each LDAP operation, use the **dirxcp** command:

[**obj**] **modify** *distinguished_name* **-attribute** *attribute_list*

Supply the distinguished name of the LDAP configuration subentry in the *distinguished_name* argument; it is **/O=My-Company/CN=ldapConfiguration** in the case of My-Company. The attribute list consists of one attribute for each type of LDAP operation; the attribute value consists of the service controls to be applied to that operation. The LDAP operation service control attributes are:

- LDAP Search Service Controls (LSES)—specifies service controls for the search operation
- LDAP Compare Service Controls (LCOS)—specifies service controls for the compare operation
- LDAP Add (LADS)—specifies service controls for the add operation
- LDAP Remove (LRES)—specifies service controls for the remove operation
- LDAP Modify (LMOS)—specifies service controls for the modify operation
- LDAP ModifyDN (LMDS)—specifies service controls for the modify DN operation

My-Company chooses the following service control strategy:

- For search and compare operations, shadowed information can be used. For the other operations, shadowed information cannot be used.
- The other service controls are the same for all operations

The following table specifies these service controls and the values selected by My-Company; see the section "LDAP Search Service Controls" in the *DirX Directory Syntaxes and Attributes* for more information about these controls.

*Table 2. My-Company LDAP Operation Service Controls*

| **Service Control** | **Possible Values** | **My-Company Value** |
|---|---|---|
| PreferChaining | **T**—prefer chaining<br>**F**—prefer referrals | **T** |
| ChainingProhibited | **T**—prohibit chaining<br>**F**—permit chaining | **F** |
| LocalScope | **T**—limit to local scope<br>**F**—no limits | **F** |
| DontUseCopy | **T**—do not use copies<br>**F**—use copies | **T (LADS, LRES, LMOS, LMDS)**<br>**F (LSES, LCOS)** |
| DontDereferenceAlias | **T**—do not dereference<br>**F**—dereference | **F** |
| Subentries | **T**—return only subentries<br>**F**—return only normal entries | **F** |

| Service Control | Possible Values | My-Company Value |
|---|---|---|
| CopyShallDo | **T**—permit partial results using shadowed information<br>**F**—prohibit partial results using shadowed information | **F (LADS, LRES, LMOS, LMDS)**<br>**T (LSES, LCOS)** |
| Priority | **0**—low (service gets low priority)<br>**1**—medium (service gets medium priority)<br>**2**—high (service gets high priority) | **1** |
| TimeLimit | *integer*—sets a limit specified by *integer*<br>**0**—no limit | **0** |
| SizeLimit | *integer*—sets a limit specified by *integer*<br>**0**—no limit | **0** |
| ScopeOfReferral | **0**—scope is DMD<br>**1**—scope is country<br>**2**—scope is unlimited | **0** |
| AttributeSizeLimit | *integer*—sets a limit specified by *integer*<br>**0**—no limit | **0** |

The following set of **dirxcp** commands defines the set of LDAP operation service controls for My-Company:

```
modify /O=My-Company/CN=ldapConfiguration -add
{LSES='preferChaining=T:chainingProhibited=F:localScope=F:dontUseCopy
=F:dontDereferenceAlias=F:subentries=F:copyShalldo=T:priori-
ty=1:timelimit=0:sizelimit=0:scopeofreferral=0:attributesizelimit=0'}
modify /O=My-Company/CN=ldapConfiguration -add
{LCOS='preferChaining=T:chainingProhibited=F:localScope=F:dontUseCopy
=F:dontDereferenceAlias=F:subentries=F:copyShalldo=T:priori-
ty=1:timelimit=0:sizelimit=0:scopeofreferral=0:attributesizelimit=0'}
modify /O=My-Company/CN=ldapConfiguration -add
{LADS='preferChaining=T:chainingProhibited=F:localScope=F:dontUseCopy
=T:dontDereferenceAlias=F:subentries=F:copyShalldo=F:priori-
ty=1:timelimit=0:sizelimit=0:scopeofreferral=0:attributesizelimit=0'}
modify /O=My-Company/CN=ldapConfiguration -add
{LRES='preferChaining=T:chainingProhibited=F:localScope=F:dontUseCopy
=T:dontDereferenceAlias=F:subentries=F:copyShalldo=F:priori-
ty=1:timelimit=0:sizelimit=0:scopeofreferral=0:attributesizelimit=0'}
modify /O=My-Company/CN=ldapConfiguration -add
```

```
{LMOS='preferChaining=T:chainingProhibited=F:localScope=F:dontUseCopy
=T:dontDereferenceAlias=F:subentries=F:copyShalldo=F:priori-
ty=1:timelimit=0:sizelimit=0:scopeofreferral=0:attributesizelimit=0'}
modify /O=My-Company/CN=ldapConfiguration -add
{LMDS='preferChaining=T:chainingProhibited=F:localScope=F:dontUseCopy
=T:dontDereferenceAlias=F:subentries=F:copyShalldo=F:priori-
ty=1:timelimit=0:sizelimit=0:scopeofreferral=0:attributesizelimit=0'}
```

#### 4.2.3.3. Granting Read Access Rights to Subentries

When the LDAP server starts, it immediately performs a special DAP bind to the contact DSA and attempts to read its configuration information from the LDAP configuration subentry. Additionally, if not prohibited, it establishes an anonymous DAP connection. It can only perform this task successfully if it the administrative point contains a subentry ACI (SACI) operational attribute that grants it read access rights to subentries. In addition, anonymous or simple authenticated LDAP v3 clients should be able to read the LDAP root subentry and the global schema, and so they must be granted read access rights to subentries as well.

Recall from the section "Bootstrapping the DSA" that the My-Company administrator created a SACI attribute that enabled him to read and modify subentries when he created the first administrative point. The My-Company administrator must now modify the first administrative point to include a new SACI that grants read-only access to subentries by unauthenticated (anonymous) users, since the LDAP server and LDAP clients will not use authentication.

The **dirxadm** and **dirxcp** stand-alone DSA setup scripts allow you to establish the SACI that grants read access rights to subentries for anonymous users when you set up the stand-alone DSA. The task is described here because it is a necessary pre-requisite for LDAP server setup and should be performed if it has not already been carried out during the stand-alone DSA setup process.

To give the LDAP server and LDAP clients read access rights to subentries, the My-Company administrator binds to the DSA, and then issues the **dirxcp** command:

```
modify /O=My-Company -addattr \
         {SACI={ID=allow anonymous to read,
                PR=10,
                AL={BL={L=NONE}},
                UF={UC={AU=TRUE},
                    UP={PR=0,PI={E=TRUE,AUATV=TRUE},
                        GAD=grantRead+grantBrowse+\
                            grantDiscloseOnError+\
                            grantReturnDN+\
```

```
                        grantFilterMatch+\
                        grantAdd+grantRemove} } } }\
        {PA={PA=My-Company Company,PA=ABC Street 123,\
             PA=D-01234 City,PA=Germany}} \
        {TN=+49 12 345 67 890} \
        {DSC=My-Company Company}
```

# 4.3. Loading the Directory Data

Now that the My-Company administrators have initialized the DBAM database, the DSA, and the LDAP server, they can proceed to load their directory data into the DBAM database.The fastest method for loading large amounts of directory data into the database is to create an LDIF content file of the directory data, and then load this file into the database with **dirxload**.A number of different LDIF file-generation tools are available (including the DSA itself); the My-Company administrators use one of these tools to create an LDIF content file of the employee data they plan to make accessible through the DirX Directory service.They name this file **Example_DB.ldif**.

The **dirxload** command for loading an LDIF file into the database is:

**dirxload –f** *LDIF_file*

The My-Company administrators run the command:

```
dirxload -f Example_DB.ldif
```

# 4.4. Starting the Service

The final step in the DirX Directory service setup process is to start the DirX Directory service.DirX Directory administrators can start (and stop) a DirX Directory service either "locally"—that is, start it from the same machine on which the service has been set up—or remotely from other machines in the enterprise network.The example discussed here assumes that the My-Company administrators are starting the service locally.

## 4.4.1. Starting a Local DirX Directory Service on Windows

To start the DirX Directory service on a Windows system you must use the Administration Tool **Services**.

## 4.4.2. Starting a Local DirX Directory Service on Linux

To start the DirX Directory service on a Linux system, use the **dirxadm start** command without any options:

```
% dirxadm start
```

The **dirxadm** program starts the local service. Note that the **dirxadm start** command does not require you to bind to the DSA first; this is the only **dirxadm** command without this requirement.

# 5. Extending the DirX Directory Service

This chapter describes how to modify the sample DirX Directory service you have established in the previous chapter (“Setting up the DirX Directory Service”). Tcl scripts for extending the service can be found in the *install_path***/scripts/stand_alone/extensions** directory and the *install_path***/scripts/security** directory. These Tcl scripts contain the commands and procedures discussed in this chapter and additional examples; view the README files in these directories for a description of their contents.

## 5.1. Extending the Stand-Alone DSA

This section of the chapter explains how you can modify the sample stand-alone DSA you have established in the previous sections.It describes how to:

- Modify the schema
- Change access control policies
- Introduce global policies to the DSA
- Add administrators who are permitted to use **dirxadm** to manage the DSA

The next sections describe the procedures to be followed to make these modifications to the sample DSA using as an example “My-Company”.The example procedures assume that the administrator has first bound to the DSA with **dirxcp** or **dirxadm**, depending on which tool should be used for the procedure.

### 5.1.1. Modifying the Schema

This section describes how to add new object classes and attributes to a DSA schema. Note that it is not possible to delete schema elements. The commands and procedures discussed in this section are contained in the in the files:

- *install_path***/scripts/stand_alone/extensions/schema_ext.cp**
- *install_path***/scripts/stand_alone/extensions/StandAloneSchemaExt.ldif**

#### 5.1.1.1. Adding New Object Classes and Attribute Types

When the DSA is initialized, it creates an LDAP global schema subentry and stores the object classes and attribute types defined in its standard schema as values of the subentry’s LOC (objectClasses) and LAT (attributeTypes) attributes. If the attribute types and object classes supplied in the standard DSA schema are insufficient to describe your DIT database, you can create new schema elements by modifying the LOC and LAT attributes in the LDAP global schema subentry. To add new object classes and attributes types to a schema, you:

1. Identify each new element with a definition and assign it at a minimum an object identifier, an LDAP name, an abbreviation, and—for attribute types—an attribute syntax. If you are creating an attribute type that is to be used in the **-filter** option of search requests or as a naming attribute, you must also specify a matching rule.

2. Modify the LDAP global schema to contain the new schema elements. You can use **dirxcp** or DirX Directory Manager to make the changes directly to the subentry, or you can put the changes into an LDIF change file and use **dirxmodify** to import them into the DIT. (Note that you cannot add to or delete the subentry; you can only modify the LOC and LAT attributes.)

The next sections describe how to accomplish these tasks using My-Company as an example.

##### 5.1.1.1.1. Identifying the New Elements

My-Company owns a car park with a number of cars, each of which can be reached by a mobile telephone. My-Company wants to add the information about the cars into its DIT as a separate organizational unit **Cars**, with the cars reflected in subordinate entries. A Car Identifier will be used to identify each car. A short description of the car will also be given as well as the telephone number at which it can be called.

To support the addition of the car information into the DIT, the administrator defines the following new schema elements:

- Two new attribute types named **car-identifier** and **car-telephone**

  The **car-identifier** attribute type has the LDAP name **carid**, the abbreviation **CARID**, the object identifier 1.2.3.4.0 and is of directory-string syntax.

  The **car-telephone** attribute type has the LDAP name **cartn**, the abbreviation **CARTN**, the object identifier 1.2.3.4.1, and has the same properties as the telephone-number standard attribute type but can contain up to 64 characters.
- One new object class named **car-id**, which has the LDAP name **car**, the abbreviation **CAR** and the object identifier 1.2.3.6.0

The object identifiers used in this example were chosen arbitrarily. In reality, they should be registered.

##### 5.1.1.1.2. Defining the New Schema Elements

You can use **dirxmodify**, **dirxcp** or DirX Directory Manager to add new schema elements to a directory schema. The **dirxmodify** and DirX Directory Manager tools use an LDAP server to make the updates to the schema, while **dirxcp** can update the schema either directly via DAP or via an LDAP server over LDAP. The next sections illustrate how to use **dirxmodify** and **dirxcp** to add the **car**, **carid** and **cartn** elements to My-Company's directory schema over LDAP.

###### Adding New Schema Elements with dirxmodify

The first step in using **dirxmodify** to update the directory schema is to create an LDIF change file with the schema elements to be added. The My-Company administrator creates an LDIF change file named **StandAloneSchemaExt.ldif** with the following content:

```
#
# DSA Schema extension
#
dn: cn=schema
changetype: modify
add: attributetypes
attributeTypes: ( 1.2.3.4.0 NAME 'carid' EQUALITY caseIg
 noreMatch ORDERING caseIgnoreOrderingMatch SUBSTR caseI
 gnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.
 15{64} )
-

dn: cn=schema
changetype: modify
add: attributetypes
attributeTypes: ( 1.2.3.4.1 NAME 'cartn' SUP telephoneNu
 mber EQUALITY telephoneNumberMatch SUBSTR telephoneNumb
 erSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.44{
 32} )
-

dn: cn=schema
changetype: modify
add: objectclasses
objectClasses: ( 1.2.3.6.0 NAME 'car' SUP top STRUCTURAL
  MUST carid MAY ( cartn $ description ) )
-
```

The LDIF change file must specify the new attribute types (**carid** and **cartn**) first, before it specifies the new object class in which they are used (**car**).

The next step is to run the following **dirxmodify** command:

**dirxmodify -f** *LDIF_file*
  **-D** *bind_DN*
  **-w** *password*
  [**-h** *host*]
  [**-p** *port_number*]

where:

- *LDIF_file* is the name of the LDIF change file

- *bind_DN* is the distinguished name of a DirX Directory administrator in LDAP format; for the My-Company administrator, the distinguished name of the default administrator account in LDAP format is **cn=admin, o=my-company**.
- *password* is the DirX Directory administrator account's password (**dirx** for the My-Company default administrator account)
- *host* is the TCP/IP address or host name of the LDAP server to which **dirxmodify** is to bind (the default is "localhost")
- *port_number* is the LDAP port number of the LDAP server to which **dirxmodify** is to bind; My-Company's LDAP server uses the default LDAP port number **389**

The My-Company administrator issues the following **dirxmodify** command:

```
dirxmodify -f StandAloneSchemaExt.ldif
           -D "cn=admin, o=my-company"
           -w dirx
```

#### Adding New Schema Elements with dirxcp

The first step in using **dirxcp** to update the directory schema is to create a project-specific abbreviation file that describes the new schema elements. The **dirxabbr** file is a standard abbreviation file that contains all the definitions, abbreviations, and object identifiers required for the standard schema of the DirX directory (see *install_path***/client/conf/dirxabbr**). The **dirxcp** and **dirxadm** tools use this file to obtain schema information when working with a DSA; LDAP clients like DirX Directory Manager and **dirxmodify** do not use this file at all.

DirX Directory supports the creation of project-specific abbreviation files to contain extensions to the standard schema. At startup, the **dirxcp** and **dirxadm** tools search for and read these files after they read the **dirxabbr** file. Project-specific abbreviation files follow the same format as the standard **dirxabbr** file and are contained in the same location. The naming convention for project-specific abbreviation files is **dirxabbr-ext-***project_identification*. The *Administration Reference* provides further details about DirX Directory abbreviation files and their format. Note that DirX Directory administrators must ensure that the LDAP names, abbreviations, and identifiers they plan to use for their schema customizations are unique over all the project-specific abbreviation files that may exist at their site.

The My-Company administrator creates a project-specific abbreviation **file dirxabbr-ext-my-company** that defines the new attribute types and object class for the car park information, as follows:

```
# Object Classes
...
CAR     Car-id  1.2.3.6.0   -   car
...
```

```
# ATTRIBUTE DEFINITION BLOCK
...
CARID   Car-Identifier  1.2.3.4.0   DIR_STR  DIR_STR    car-id
CARTN   Car-Telephone       1.2.3.4.1   PRINTABLE_STR
        PRINTABLE_STR   car-tn
...
```

The next step is to issue a **dirxcp bind** command over LDAPv3, then use **dirxcp** command:

[**obj**] **modify** *distinguished_name* **-addattr** *attribute_list*

where *distinguished name* is the name of the schema subentry—in this case, **CN=schema**—and *attribute_list* contains the new object types and attribute types to be added to the subentry's LOC (objectClasses) and LAT (attributeTypes) attributes.

My-Company uses the following set of **dirxcp** commands to define its new elements:

```
modify cn=schema -addattr {attributeTypes= \
    ( 1.2.3.4.0 NAME 'carid' \
     DESC 'My-Company car identifier' \
     EQUALITY caseIgnoreMatch \
     ORDERING caseIgnoreOrderingMatch \
     SUBSTR caseIgnoreSubstringsMatch \
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.15{64} )}
```

This **dirxcp** command adds the attribute type car-identifier (**carid**) to the schema. The attribute syntax (SYNTAX) is DirectoryString (object identifier **1.3.6.1.4.1.1466.115.121.1.15**) with a length of up to 64 characters. The Equality-, Ordering- and Substrings-Matching-Rules (EQUALITY, ORDERING, SUBSTR) are caseIgnoreMatch, caseIgnoreOrderingMatch and caseIgnoreSubstringsMatch. Refer to RFC 2252 for further details about LDAP attribute type and object class descriptions.

```
modify cn=schema -addattr {attributeTypes= \
    ( 1.2.3.4.1 NAME 'cartn' \
     DESC 'My-Company car telephone number' \
     SUP telephoneNumber
     EQUALITY telephoneNumberMatch \
     SUBSTR telephoneNumberSubstringsMatch \
     SYNTAX 1.3.6.1.4.1.1466.115.121.1.44{32} )}
```

This **dirxcp** command adds the attribute type car-telephone (**cartn**) to the schema. The attribute is a subtype (SUP) of the standard attribute Telephone-Number (telephoneNumber). The attribute syntax (SYNTAX) is Printable-String (whose object

identifier is 1.3.6.1.4.1.1466.115.121.1.44) with a maximum length of 64 characters. The Equality- and Substrings-Matching-Rules are the same as for the supertype; if not specified, they are automatically derived from the supertype.

```
modify cn=schema -addattr {objectClasses= \
    ( 1.2.3.6.0 NAME 'carid' \
     DESC 'My-Company car identifier object class' \
     SUP top AUXILIARY \
     MUST carid MAY ( description $ cartn ) )}
```

This **dirxcp** command adds an auxiliary object class car-id (**car)** with a car-identifier attribute (**carid**) as a mandatory attribute and a Description (description) and **cartn** telephone-Number (telephoneNumber) as optional attributes.

##### 5.1.1.1.3. Propagating Schema Updates to Multiple LDAP Servers

The directory schema is stored in the DIT. LDAP servers read the directory schema when they start up to obtain the information about object classes and attribute types contained within it. When you update the LDAP global schema subentry with a DirX Directory administration tool over LDAP, the LDAP server that performs the update request automatically reads the updated schema and thus synchronizes its schema knowledge with the directory schema's current state. If your configuration contains multiple LDAP servers, you will need to restart them by hand so that they will re-read the updated schema.

##### 5.1.1.1.4. Populating the Directory with the New Schema Objects

Once you have modified the directory schema with new or modified schema elements, you can then use **dirxcp**, DirX Directory Manager or **dirxmodify** to populate the directory database with objects that conform to the updated schema.

For example, the My-Company administrator can use the following **dirxcp obj create** commands to create the car organizational unit and a car identifier entry (after first performing a DAP bind to the DSA as the default administrator):

```
create O=My-Company/OU=Cars -attr {OCL=TOP;OU}
create O=My-Company/OU=Cars/CARID=ABC-1234 -attr \
       {OCL=TOP;CAR} \
       {DSC=Red Italian Car} \
       {CARTN=+12 34 567 893}
```

### 5.1.2. Modifying Access Control Policies

The initial access control policy that the administrator established when it set up its stand-alone DSA allows only the DSA administrator to modify any data in the DIT. Suppose the administrator now wants to modify this access control policy to permit users binding with simple authentication and a password to change their passwords and their telephone

numbers.

Recall that the prescriptive-ACI (PACI) attribute of the access control subentry specifies the access rights of different users. When the administrator created the access control subentry during stand-alone DSA setup, he specified two types of users in the PACI attribute: the administrator and anonymous users. To establish the new access control policy granting users the rights to change their passwords and telephone numbers, the DSA administrator must add a new prescriptive-ACI (PACI) operational attribute value for the user type “My-Company users” to the access control-specific subentry that specifies the desired access rights.

To add attributes to an entry or subentry with **dirxcp**, you first issue a **dirxcp bind** command over DAP. You then use the **dirxcp** command:

[**obj**] **modify** *distinguished_name* **-addattr** *attribute_list*

The following **dirxcp** command adds the new PACI attribute value to the My-Company DSA’s access control-specific subentry:

```
modify /O=My-Company/CN=AccessControl-Subentry -addattr \
{PACI={ID=My-Company users: can modify their passwords and tele-phone
numbers,
       PR=100,
       AL={BL={L=SIMPLE}},
       UF={UC={TH=TRUE},
           UP={PI={E=TRUE},
              GAD=grantModify
              };
              {PI={AT=UP;TN,
                   AAV=UP;TN},
               GAD=grantAdd+grantRemove
              }
          }
      }
}
```

This command adds a new PACI attribute value to the existing access control-specific subentry. The user-first (UF) component specifies in the user-class (UC) that the user whose name matches that of the entry being accessed (this-entry (TH)) is given the rights of modifying the entry (adding and deleting the attributes user-password (UP) and telephone-number (TN)).

### 5.1.3. Creating Collective Attributes in a Subtree

Collective attributes are used to specify attributes that are common to all entries of a subtree within the administrative area. The values of collective attributes are returned with

the results of read or search operations. The DSA does not support collective attributes in filters.

To create collective attributes in a subtree, carry out the following steps:

1. Define a collective attribute-specific administrative point
2. Create a collective attribute-specific subentry

The next sections describe how to perform each of these tasks using My-Company as an example. My-Company's telephone operator can be reached through a central telephone number. My-Company wants this telephone number to appear for all employees and so plans to add it as a collective attribute.

#### 5.1.3.1. Defining the Collective Attribute-Specific Administrative Point

To permit collective attributes to be used below a particular entry, the entry must be defined as a collective attribute-specific administrative point. This is achieved by adding the value collective-attribute-specific-area (CASA) to the administrative-role (AR) attribute of the entry. If the DSE-Type of the entry has not been administrative-point (ADM_POINT) before this modification, this value is added automatically.

To permit collective attributes for an entry, use the **dirxcp** command:

**modify** *distinguished_name* **-addattr {AR=CASA}**

For the company, a collective telephone number is to be added to the entry for the My-Company organization. This entry is already an access-control-specific administrative point, but it is currently not allowed to contain a collective attribute-specific subentry. Consequently, the default administrator must modify the administrative role of the My-Company organization entry (which is the first administration point) using the **dirxcp** command:

```
modify /O=My-Company -addattr {AR=CASA}
```

### 5.1.4. Adding DirX Directory Administrators to a DSA

The My-Company default administrator is currently the only user who can use **dirxadm** (or DirX Directory Manager) to bind to and access the entries in the My-Company stand-alone DSA. The administrator hires a second-shift administrator, to whom he wants to give the same access rights to manage the DSA in his absence. To set up the DSA to be managed by this new administrator, the administrator must follow these steps:

1. Create an entry in the DIT for the second-shift administrator
2. Add the second-shift administrator's distinguished name to the DirX-Administrators attribute
3. Modify the subentry ACI of the administration point to permit the second-shift administrator to handle subentries

4. Modify the prescriptive ACI of the access-control-specific subentry to permit the same access rights to entries for the second-shift administrator

The administrator can perform these tasks with DirX Directory Manager or with **dirxcp** and **dirxadm**. The next sections illustrate how to perform these tasks with **dirxcp** and **dirxadm**.

#### 5.1.4.1. Adding the New Administrator Entry

The My-Company administrator decides to establish the convention of starting all DirX Directory administrator common names with **admin**; that is, their common name must start with the form **cn=admin**. The administrator plans to use the common name **cn=admin2** for the second-shift administrator.

To create the entry for the second-shift administrator in the DIT, the administrator binds to the DSA with **dirxcp** over LDAPv3, and then issues the **dirxcp** command:

```
create cn=admin2,o=My-Company \
       -attr   {objectClass=top;person;organizationalPerson} \
               {sn=admin2} \
               {description=Second-shift Administrator} \
               {userPassword=dirx2}}
```

The command adds the second-shift administrator entry to the DIT with the distinguished name **cn=admin2,o=My-Company**.

#### 5.1.4.2. Adding the New Administrator to the DADM Attribute

To add the administrator's distinguished name to the DirX Administrators attribute, the My-Company administrator binds to the DSA with **dirxadm** and issues the command:

```
modify / -addattr DADM={/O=My-Company/CN=admin2}
```

The My-Company second-shift administrator can now use **dirxadm** to bind to the DSA with simple authentication and the password **dirx2**.

#### 5.1.4.3. Granting the New Administrator Access Rights to Subentries

To give the second-shift administrator the same rights to subentries, the My-Company administrator binds to the DSA with **dirxcp** over DAP (using LDAP syntax to specify an ACI is very cumbersome) and issues the command:

```
modify /O=My-Company -addattr \
         {SACI={ID=admin2: enable Handling of Subentries,
                PR=254,
                AL={BL={L=SIMPLE}},
```

```
                UF={UC={N={DN={/O=My-Company/CN=admin2}}},
                    UP={PI={E=TRUE},
                        GAD=grantRead+grantBrowse+\
                            grantDiscloseOnError+\
                            grantReturnDN+\
                            grantAdd+grantRemove+grantModify};
                       {PI={AT=SS;OC;AT;DSR;
                            DCR;MR;MRU;CRN;CRT;MN;MT;PACI;EACI,
                            AAV=SS;OC;AT;DSR;
                            DCR;MR;MRU;CRN;CRT;MN;MT;PACI;EACI,
                            AUATV=TRUE},
                        GAD=grantDiscloseOnError+\
                            grantRead+grantCompare+\
                            grantFilterMatch+\
                            grantAdd+grantRemove} } } } \
          {PA={PA=My-Company Company,PA=ABC Street 123,\
               PA=D-01234 City,PA=Germany}} \
          {TN=+49 12 345 67 890} \
          {DSC=My-Company Company}
```

#### 5.1.4.4. Granting the New Administrator Access Rights to Entries

To give the second-shift administrator the same access rights to the entries in the DIT, the My-Company administrator binds to the DSA with **dirxcp** over DAP and issues the command:

```
modify /O=My-Company/CN=AccessControl-Subentry  -addattr \
           {PACI={ID=admin2: enable most of operations \
                     but no Rename,
                 PR=254,
                 AL={BL={L=SIMPLE}},
                 UF={UC={N={DN={/O=My-Company/CN=admin2}}},
                     UP={PI={E=TRUE},
                     GAD=grantDiscloseOnError+\
                         grantRead+grantBrowse+\
                         grantReturnDN+\
                         grantAdd+grantRemove+grantModify};
                     {PI={AT=AR;ACS;SACI;CRN;
                          CRT;MN;MT;SOC;GSR;CE;EACI,
                          AAV=AR;ACS;SACI;CRN;
                         CRT;MN;MT;SOC;GSR;CE;EACI,
```

```
                              AUATV=TRUE},
                         GAD=grantDiscloseOnError+\
                             grantRead+grantCompare+\
                             grantFilterMatch+\
                             grantAdd+grantRemove}

 } };
```

# 5.2. Extending the My-Company LDAP Server Configuration

This section explains how to modify the sample LDAP server configuration established in “Setting up the DirX Directory Service”.It describes how to:

- Change the values in the LDAP configuration subentry
- Set up the LDAP server to support SSL/TLS connections
- Set up and manage multiple LDAP servers
- Activate configuration changes to multiple LDAP servers
- Use IP addresses to control client access to the LDAP server
- Modify the default LDAP cache configuration
- Add user policies to an LDAP server

The next sections describe the procedures to be followed to make these modifications to the sample configuration using the My-Company example.

## 5.2.1. Changing the LDAP Configuration

To change an LDAP server’s configuration, you must modify the appropriate attributes of its LDAP server configuration subentry. For example, suppose the My-Company administrator wants to prohibit chaining for the LDAP modifyDN operation. To enable this configuration, he can use DirX Directory Manager or **dirxcp** over LDAPv3 to modify the corresponding attribute in the LDAP server’s configuration subentry to prohibit chaining.

To change attribute values in an LDAP configuration subentry with **dirxcp**, bind to the LDAP server’s contact DSA with **dirxcp** over LDAPv3, then use the command:

[**obj**] **modify** distinguished_name **-replaceattr** attribute_list

where *distinguished_name* is the name of the subentry. In the case of My-Company, the distinguished name (in LDAP syntax) is **cn=ldapConfiguration,o=My-Company**. To change the service control for the LDAP modifyDN operation to prohibit chaining, the My-Company administrator issues the following **dirxcp** command:

```
modify cn=ldapConfiguration,o=My-Company -replaceattr
```

```
{ldapModifyDNSvcCtl='preferChaining=T:
chainingProhibited=T:
localScope=F:dontUseCopy=T:
dontDereferenceAlias=F:
subentries=F:
copyShalldo=F:
priority=1:
timelimit=0:
sizelimit=0:
scopeofreferral=0:
attributesizelimit=0'}
```

## 5.2.2. Activating Configuration Changes in the LDAP Server

There are two methods for activating changes to LDAP server configuration attributes:

- Activating the change dynamically with **dirxextop** or DirX Directory Manager
- Activating the change by re-starting the LDAP server

### 5.2.2.1. Activating Configuration Changes Dynamically

The DirX Directory service permits a subset of LDAP configuration subentry attributes to be dynamically updated, which preserves existing connections from LDAP clients to the LDAP server instead of losing them due to LDAP server process exit on a re-start. To activate changes made to these attributes dynamically, you can use the **dirxextop** command with the **ldap_cfg_upd** extended operation or the following DirX Directory Manager path:

**Monitoring** → **LDAP** → **Configuration** → **Update CFG attributes**

In the previous step, the My-Company administrator changed the service control for the LDAP modifyDN operation to prohibit chaining. Changes to this service control are allowed to be dynamically activated. To activate the change dynamically with **dirxextop**, the My-Company administrator issues the following command:

```
dirxextop -D cn=admin,o=my-company -w dirx -t ldap_cfg_upd
```

For the list of LDAP configuration subentry attributes that can be updated dynamically, see the description of the **ldap_cfg_upd** extended operation in the *DirX Directory LDAP Extended Operations*.

### 5.2.2.2. Activating Configuration Changes with Server Re-start

Some LDAP configuration subentry attributes cannot be changed dynamically; for example, the LDAP port number (LPNU) user attribute. Changes to these attributes can only become effective after a restart of the LDAP server.

To restart the LDAP server, bind to the DSA with **dirxadm dse bind** (see *Administration Reference* for details) and use the command:

**ldap restart**

This command stops and restarts the LDAP server and activates a change, for example, to the LDAP port number (LPNU) user attribute as a result of the re-start.

Note that the re-start process disconnects all LDAP client connections and causes the server to re-read the directory schema.

## 5.2.3. Enabling SSL/TLS Support

Recall from the section “DirX Directory Security Services” in the *DirX Directory Introduction* that DirX Directory supports the Secure Socket Layer/Transport Layer Security (SSL/TLS) protocols for encrypted and authenticated LDAP client-server communications over the Internet and other TCP/IP networks. SSL/TLS session encryption ensures LDAP traffic confidentiality at the network level. When My-Company established its DirX Directory service (as described in the chapter “Setting up the DirX Directory Service), it chose not to enable SSL/TLS for its LDAP server.

Now My-Company wants to make its employee credit card numbers available to the Frankfurt travel agency for use in making travel arrangements on the employees' behalf. To protect the exchange of this information, they decide to enable the LDAP server to accept SSL connections from the Frankfurt travel agency. The company decides not to require that client authentication be performed over the SSL-encrypted transmission.

Setting up the LDAP server to support the SSL protocol consists of the following steps:

1. Adding SSL-specific information to the LDAP configuration subentry
2. Creating the SSL configuration subentry

You can perform these steps with DirX Directory Manager or with **dirxcp**. The next sections describe these steps and how to perform them with **dirxcp**.

### 5.2.3.1. Adding SSL Information to the LDAP Configuration Subentry

The LDAP configuration subentry contains three SSL-specific attributes:

- The LDAP secure port number LSPN (ldapSecurePortNumber) user attribute, which specifies the secure port number on which the LDAP server is to listen for SSL/TLS requests from LDAP clients.

An LDAP server that supports SSL/TLS protocol needs to be assigned a port on which to listen for incoming requests over SSL/TLS from LDAP clients. In general, LDAP servers are assigned the secure port number **636**. In some cases, however, this port may be in use by another service, and so a different LDAP server port must be selected. Since the machine on which the LDAP server is to run has nothing assigned to port **636**, My-Company decides to use this port number for its LDAP server.

- The LDAP SSL configuration subentry common name LSCCN

(ldapSSLCfgSubschemaCN) attribute, which specifies the value of the common name assigned to the LDAP SSL configuration subentry.

An LDAP SSL configuration subentry specifies the configuration parameters for SSL/TLS support. There can be more than one LDAP SSL configuration subentry per DSA; this is so if the configuration supports multiple LDAP server processes that are to have different SSL settings. The administrator of the contact DSA must create the SSL configuration subentry below the first context prefix using **dirxcp** over LDAPv3; the common name applied to this subentry must be the same as the value of the LSCCN attribute. The default name for the SSL configuration subentry is **ldapSSLConfiguration**; My-Company chooses to use this default.

- The LDAP StartTLS enabled LSTEN (ldapStartTLSenabled) user attribute, which specifies whether a plain LDAP connection can be switched to SSL/TLS with the startTLS extended operation. By default, the support of startTLS is disabled. My-Company decides to use this default behavior of the plain LDAP port.

##### 5.2.3.1.1. Adding SSL-Specific Information

To modify the LDAP configuration subentry with **dirxcp**, the My-Company administrator binds to the contact DSA with **dirxcp** over LDAPv3 and uses the command:

[**obj**] **modify** *distinguished_name* **-replaceattr** *attribute_list*

where *distinguished_name* is the name (in LDAP syntax) of the My-Company subentry **cn=ldapConfiguration,o=My-Company**. To add the SSL-specific information to the subentry, the My-Company administrator issues the following **dirxcp** command:

```
modify cn=ldapConfiguration,o=My-Company \
    -replaceattr {ldapSecurePortNumber=636} \
                 {ldapSSLCfgSubschemaCN=ldapSSLconfiguration}
```

#### 5.2.3.2. Creating the SSL Configuration Subentry

The LDAP SSL configuration subentry contains attributes that control how SSL functions, including:

- The cipher suites (signature, encryption, and hashing algorithms) that the LDAP server supports for the SSL/TLS protocol handshake (attribute LSSES (ldapSupportedEncryptionStrength) and attribute LSSESEX (ldapSupportedEncryptionStrengthExt )).
- The security protocols that the LDAP server supports: TLS v1.0, v1.1, v1.2 or v1.3) (attribute LSSSP (ldapSupportedSecurityProtocols)).
- The public key/private key material that has been assigned to the LDAP server: its user certificate and its private key (attribute LSOKP (ldapOwnKeymaterialPEM)). This attribute must contain the data in password-protected Privacy Enhanced Mail (PEM) format. The LDAP server uses this key material to authenticate itself to a client.

- The public key material of the Root Certification Authorities that the LDAP server trusts to issue correct client certificates (attribute LSTCC (ldapTrustedCaCerts)). The LDAP server uses this key material to verify the client's authentication. This attribute must contain the binary public key certificates of the trusted CAs. Note that this key material is only needed if client authentication is required.
- The files that contain the Certificate Revocation Lists (CRLs) that the LDAP server is to use to check the user certificates presented during certificate-based client authentication and when executing options (specified via other subentry attributes) that control the extent of CRL checking (attributes LSCRLC (ldapSslCrlChecking), LSTMCRL (ldapSslTolerateMissingCrl), LSANCRL (ldapSslAllowNYVCrl), LSAECRL (ldapSslAllowExpiredCrl)). The files must contain CRLs issued by the trusted CAs in PEM format (attribute LSCRLF (ldapSslCrlFileNames)).
- The file that contains the password that the LDAP server is to use to access its own key material (attribute LSPPF (ldapPkcs12PasswordFile)).
- The way in which client authentication is mapped onto a directory DN (attributeLSAIM ldapSaslAuthzIdMapping).
- Whether client authentication is required during the SSL handshake (attribute LSCAR (ldapSSLClientAuthRequired)).
- Configuration parameters for SSL event logging (attributes LSTFI and LSTLE).

The following figure illustrates the relationship between the LDAP configuration and the LDAP SSL configuration subentries and the relationship between the LDAP SSL configuration subentry and the files for SSL support. The figure assumes that client authentication is not required.

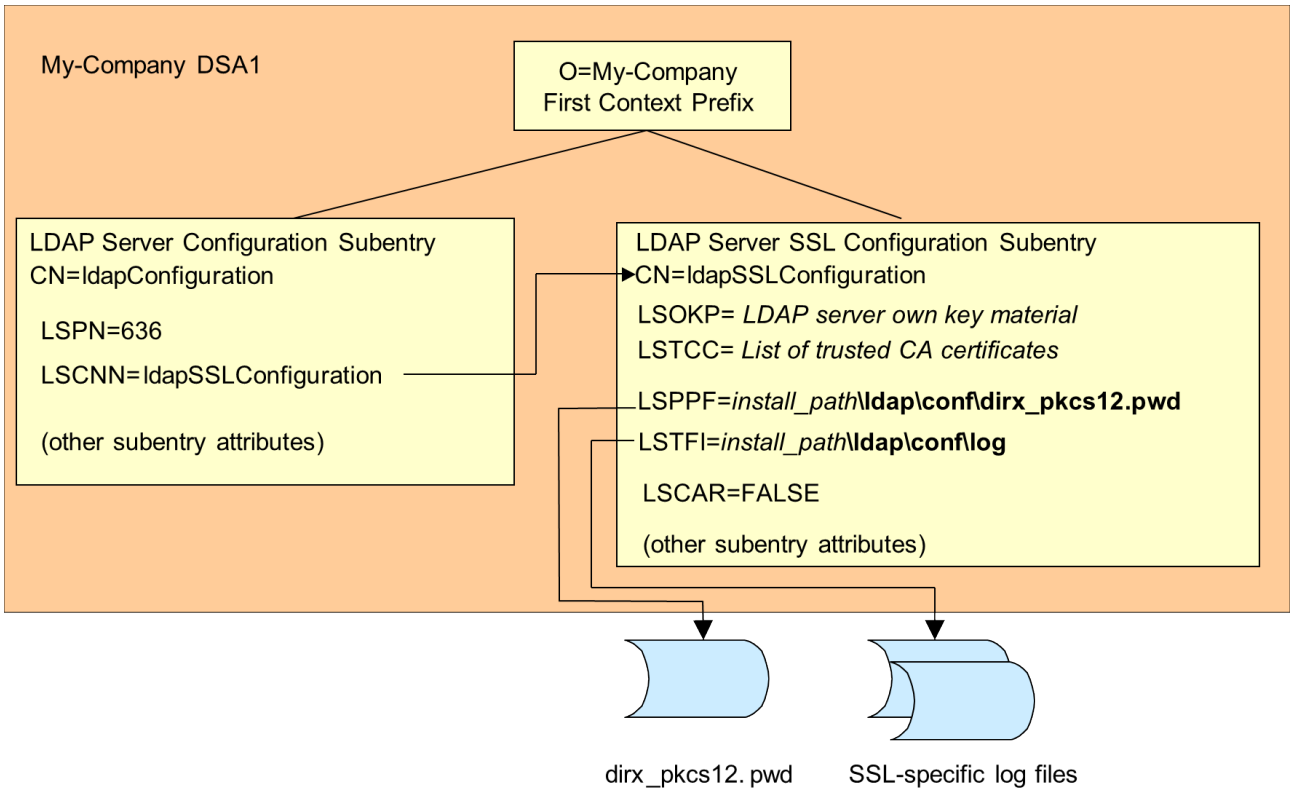



*Figure 26. LDAP Server Configuration and LDAP Server SSL Configuration*

The content of the key material depends on the Public Key Infrastructure (PKI) in use. Among other things, the decision about which certificate issuers are to be trusted by LDAP clients and the LDAP server is the result of the security policy planning process. This task is separate from the task of setting up an LDAP server.

#### 5.2.3.2.1. Creating the LDAP SSL Configuration Subentry

To create the LDAP SSL configuration subentry below the first context prefix (which in the case of My-Company is also the first administrative point), bind to the contact DSA with **dirxcp** over DAP, and then use the command:

[**obj**] **create** *distinguished_name* **-attribute** *attribute_list*

Supply the distinguished name of the LDAP SSL configuration subentry in the *distinguished_name* argument; it is **/O=My-Company/CN=ldapSSLConfiguration** (DAP syntax) in the case of My-Company. The *attribute list* parameter can contain one or more SSL configuration attributes. These attributes are optional; if an SSL configuration attribute is not supplied, the DirX Directory LDAP server uses a default value.

For My-Company, the attribute list contains the following SSL configuration attributes:

- The LDAP supported security protocols (LSSSP) user attribute. My-Company selects **TLSv1.2 and TLSv1.3** (by default, the LDAP server accepts all versions of the TLS protocol).
- The LDAP supported encryption strength (LSSES) user attribute, which specifies the ciphers that are accepted in the SSL handshake protocol for protocol versions lower than TLSv1.3. My-Company selects **HIGH** (the default value is **RSA** (all cipher suites that use the RSA algorithm are accepted)).
- The LDAP supported encryption strength extended (LSSESEX) user attribute, which specifies the cipher suites that are accepted in the SSL handshake protocol for the protocol version TLSv1.3. My-Company selects **TLS_AES_256_GCM_SHA384**.
- The LDAP PKCS12 password file (LSPPF) user attribute, which specifies the full pathname of the LDAP server's key material password file. My-Company selects **C:\Program Files\DirX\Directory\ldap\conf\dirx_pkcs12.pwd**.
  The content of the dirx_pkcs12.pwd file is the clear text password only after the initial administration. Once the LDAP server has accessed the file one time, it changes the content to the encrypted format of the password.
- The LDAP server key material PEM (LSOKP) user attribute, which contains the LDAP server's key material. My-Company provides sample key material in the file **C:\Program Files\DirX\Directory\ldap\conf\cert_ldapserver.pem** and loads the file content into the attribute with the **dirxcp create** operation.
- The LDAP SSL trace file (LSTFI) user attribute, which specifies the full pathname of the directory where the SSL-specific log files are located. My-Company selects **C:\Program Files\DirX\Directory\ldap\log** (which is the default value on Windows systems).
- The LDAP SSL client authentication required (LSCAR) user attribute, which specifies whether the LDAP server will enforce SSL client authentication. My-Company chooses **FALSE**, which is the default.

- The LDAP SSL trace level (LSTLE) user attribute, which specifies the level of SSL internal logging. My-Company chooses the default value **0**, which turns off trace logging. It is recommended to turn on SSL internal logging only if TLS handshake problems must be debugged. The logging can result in huge logging files that may contain confidential data.

The attribute list must also contain at least the following subentry-specific attributes:

- The object class (OCL) user attribute that specifies the Subentry (SUBE) and LDAP configuration (LCFG) object classes.
- The subtree-specification (SS) operational attribute, which must specify its default value, indicating that the subentry holds for the entire subtree.

The following **dirxcp** command creates an LDAP SSL configuration subentry for My-Company:

```
  create /O=My-Company/CN=ldapSSLConfiguration -attr \
         {OCL=SUBE;LSCFG} \
         {SS={DEF=TRUE} \
         {LSSSP=TLSv1.2;TLSv1.3} \
         {LSSES=HIGH} \
         {LSSESEX=TLS_AES_256_GCM_SHA384} \
         {LSPPF=
C:\\Program Files\\DirX\\Directory\\ldap\\conf\\dirx_pkcs12.pwd} \
         {LSOKP_FILE=
C:\\Program Files\\DirX\\Directory\\ldap\\conf\\cert_ldapserver.pem}\
         {LSTFI=C:\\Program Files\\DirX\\Directory\\ldap\\log} \
         {LSCAR=FALSE}
```

The backslash (\) character must be escaped by a backslash (\) character.

### 5.2.4. Setting up Multiple LDAP Servers

In some cases, DirX Directory administrators may want to run several LDAP servers on a single machine. Possible reasons can include:

- When the LDAP clients using the LDAP service handle character sets in different ways (LDAP v2 Latin1, T61)
- When administrators want to have special LDAP servers where no modifications to the database are allowed (Read-Only)
- When administrators want to set up an extra LDAP server that is only accessible via SSL

All of the LDAP servers communicate with the same DSA—the contact DSA specified in the **dirxldap.cfg** file—and are defined by the same LDAP root subentry, but they must listen for LDAP client requests on different ports, and are configured to handle a particular type of

LDAP client. If both LDAP servers are assigned to support LDAP over SSL/TLS, their secure port numbers (that is, the values of their LSPN attributes) must be different.

In addition, each additional LDAP server must be assigned its own RPC port number on which to listen for **dirxadm** administrative requests. The primary LDAP server—which is the first server to be set up—establishes an RPC port mapper (necessary for dirxadm RPC commands) at port 6999 by default. Any additional LDAP servers that are set up must be assigned different RPC port numbers. DirX Directory administrators must use the RPC port number as input to the **dirxadm ldap binding** operation, which redirects subsequent **dirxadm ldap** operations to the LDAP server listening on the RPC port number specified in the operation. In this way, they can switch between administration of the multiple LDAP servers.

The My-Company administrator decides that his LDAP server configuration requires a second LDAP server that supports the T61 character set. He also decides that the additional LDAP server does not need to support SSL/TLS. Before he can set up this second LDAP server, he must select:

- A different free LDAP request port number for the second LDAP server, the LDAP request port number he selects is **588**.
- A different common name to assign to the new server's LDAP configuration subentry. He selects the common name **ldapConfig2** for the second server's LDAP configuration subentry.
- A different unused RPC port on which the new server is to listen for **dirxadm** requests. The RPC port number he selects is **8999** (he uses the **netstat -a** command to determine whether the port he has selected is in use by another process).

  The following figure illustrates My-Company's multiple LDAP server configuration.

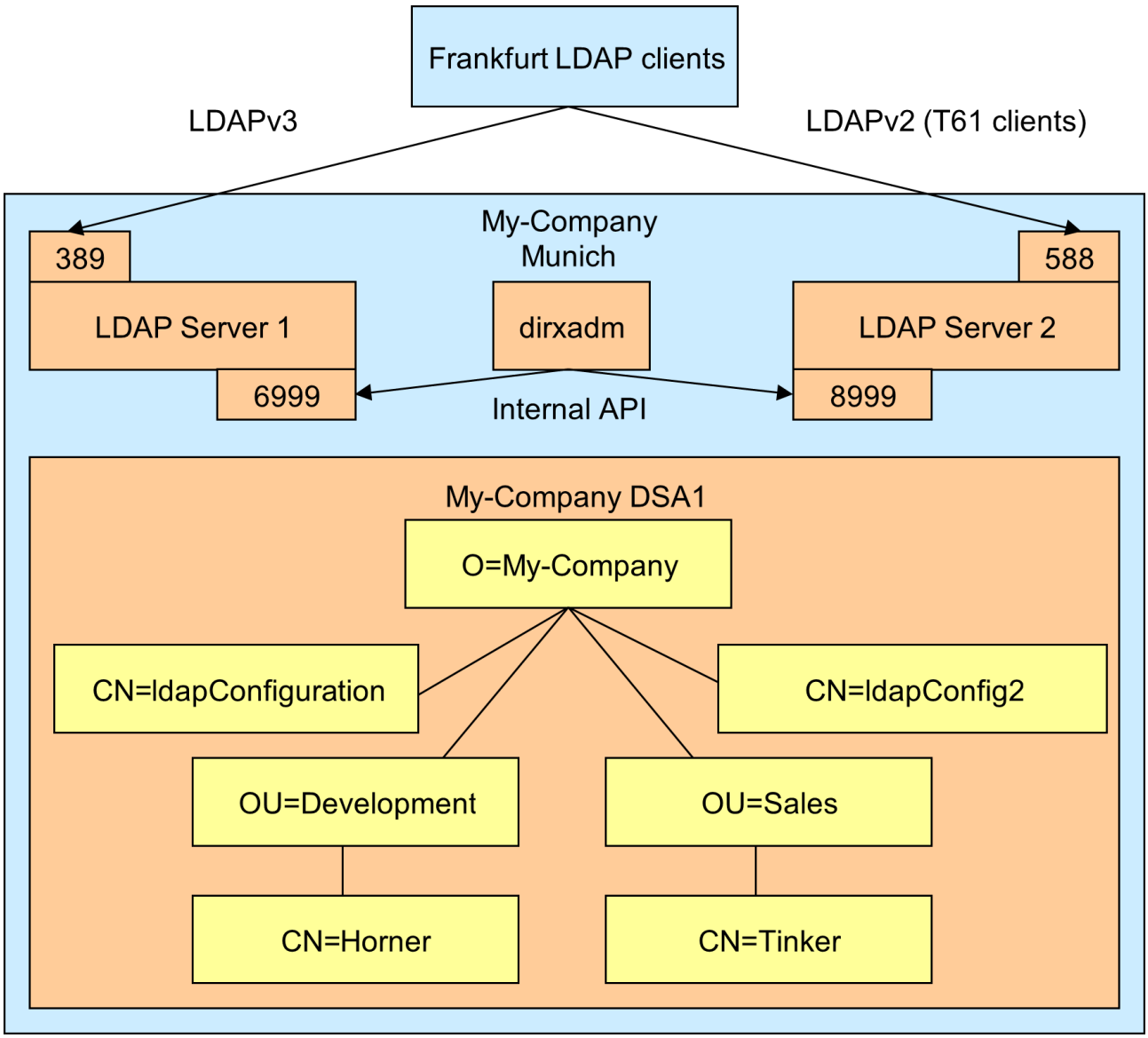



*Figure 27. My-Company Multiple LDAP Server Configuration*

Next, the My-Company administrator must:

- Create a new LDAP configuration subentry for the additional server
- Start the new LDAP server

The next sections describe how to perform these tasks with **dirxcp** and **dirxadm**.

#### 5.2.4.1. Creating the LDAP Configuration Subentry

To create the LDAP configuration subentry, bind to the contact DSA with **dirxcp** over DAP and then use the **dirxcp** command:

[**obj**] **create** *distinguished_name* **-attribute** *attribute_list*

Supply the distinguished name of the new LDAP configuration subentry in the *distinguished_name* argument; My-Company chooses **/CN=ldapConfig2**. My-Company supplies the following in *attribute_list*:

- The LDAP port number (LPNU) user attribute with the value **588**.
- The LDAP Character Set Conversion Request (LCCQ) user attribute with the value **T61**.
- The LDAP Character Set Conversion Result (LCCS) user attribute with the value **T61**.
- For the other attributes, the same values as specified for the **ldapConfiguration** subentry.

The following **dirxcp** command creates the second LDAP configuration subentry for My-Company:

```
create /O=My-Company/CN=ldapConfig2 -attr \
       {OCL=SUBE;LCFG} \
       {SS={DEF=TRUE}} \
       {LPNU=588} \
       {LSPN=0} \
       {LMCO=1024} \
       {LCIT=300} \
       {LUDT=0} \
       {LAHE=FALSE} \
       {LROS=FALSE} \
       {LCCQ=T61} \
       {LCCS=T61}
```

#### 5.2.4.2. Starting the Second LDAP Server

Starting the second LDAP server consists of two steps:

- Setting the **DIRX_ADDITIONAL_LDAP_SERVERS** environment variable
- Restarting the directory service

The **DIRX_ADDITIONAL_LDAP_SERVERS** environment variable specifies the LDAP configuration subentry name and the RPC port number of each additional LDAP server that is to be started as part of the directory service. The procedure to set this environment variable depends on the operating system in use.

After you have set the **DIRX_ADDITIONAL_LDAP_SERVERS** environment variable, you must restart the directory service. This step starts all of the DirX Directory servers, including the additional LDAP servers specified in the **DIRX_ADDITIONAL_LDAP_SERVERS** environment variable.

##### 5.2.4.2.1. Starting the Second LDAP Server on Windows Systems

On Windows systems, you access the **DIRX_ADDITIONAL_LDAP_SERVERS** environment variable with the path **Start** → **Settings** → **System** → **About** → **System info** → **Advanced system settings** → **Advanced** tab. The value of the environment variable is the common name of the LDAP configuration subentry that corresponds to the new LDAP server and its

RPC port number, in the format *subentry_name+RPC_port_number*[: ...]. If you are starting more than one additional LDAP server, use the colon as a separator between subentry+RPC_port pairs. For example, the My-Company administrator enters **ldapConfig2+8999**.

After you have set this environment variable, restart the system. An LDAP server that reads the configuration subentry with the common name **ldapConfiguration** and an LDAP server that reads the LDAP configuration subentry with the common name **ldapConfig2** are started.

##### 5.2.4.2.2. Starting the Second LDAP Server on Linux Systems

To set the **DIRX_ADDITIONAL_LDAP_SERVERS** environment variable on Linux systems, you edit the file *install_path/***.dirxrc**, and include two lines that set the environment variable, in the format:

**DIRX_ADDITIONAL_LDAP_SERVERS=***subentry_name*+_RPC_port_number_[: ...]
**export DIRX_ADDITIONAL_LDAP_SERVERS**

where *subentry_name* is the common name of the LDAP configuration subentry that corresponds to the new LDAP server and RPC_port_number is the RPC port number assigned to the new LDAP server. (If you are starting more than one additional LDAP server, use the colon as a separator between subentry+RPC port pairs.) For example, the My-Company administrator adds the following lines to the **dirxrc** file:

**DIRX_ADDITIONAL_LDAP_SERVERS=ldapConfig2+8999**
**export DIRX_ADDITIONAL_LDAP_SERVERS**

After you have set the environment variable:

1. Bind to the contact DSA with **dirxadm bind** (if you have not done so already).
2. Issue the **dirxadm** command **sys stop** to stop the running DSA and LDAP server.
3. Issue the **dirxadm** command **sys start** to restart the contact DSA and LDAP server. An LDAP server that reads the configuration subentry with the common name **ldapConfiguration** (the default common name) and an LDAP server that reads the LDAP configuration subentry with the common name **ldapConfig2** are started.

## 5.2.5. Managing Multiple LDAP Servers

DirX Directory administrators can use the **dirxadm ldap binding** operation with the **-port** option to move between the management of multiple LDAP servers that are running on the local system (they can use the **-host** and **-port** options to manage remote LDAP servers). For example, suppose the administrator for the travel agency in Frankfurt reports that its LDAP clients are having problems when they attempt to connect to the My-Company database. The My-Company administrator decides to enable diagnostic logging for both LDAP servers so that he can try to detect the problem. To enable diagnostic logging for both LDAP servers, the My-Company administrator performs the following sequence of **dirxadm** operations:

1. Binds to the contact DSA with the **bind** operation. This command directs **dirxadm** to

send administrative requests to port **6999**, which is the default RPC port mapper of the LDAP server and is also the first LDAP server's RPC port.

2. Issues the **ldap log** operation, which enables diagnostic logging for the first LDAP server.
3. Issues the **ldap binding -port 8999** operation to redirect **dirxadm** administrative requests to the RPC port of the second LDAP server.
4. Issues the **ldap log** command, which starts diagnostic logging on the second LDAP server.

## 5.2.6. Activating Configuration Changes in Additional LDAP Servers

Activating configuration changes in any additional LDAP servers consists of two steps:

1. Using the **ldap binding** operation with the **-port** option to redirect **dirxadm ldap** operations to the RPC port mapper of the target LDAP server
2. Activating the configuration change, either dynamically or with an LDAP server re-start

### 5.2.6.1. Activating Configuration Changes Dynamically

Recall from the section "Changing the LDAP Configuration" that the DirX Directory service permits a subset of LDAP configuration subentry attributes to be dynamically updated, which preserves existing connections from LDAP clients to the LDAP server instead of losing them due to LDAP server process exit on a re-start. As described in that section, the My-Company administrator changed the service control for the LDAP modifyDN operation to prohibit chaining.

Now the My-Company administrator wants to duplicate this change on the second LDAP server (see "Setting up Multiple LDAP Servers"). Recall from the earlier section that changes to this service control are permitted to be dynamically activated with the **dirxextop** command or with DirX Directory Manager. To activate the change dynamically with **dirxextop** on the second LDAP server, the My-Company administrator issues the following command:

```
dirxextop -D cn=admin,o=my-company -w dirx -t ldap_cfg_upd
-h /CN=DirX-DSA-MyHostname -p 8999
```

### 5.2.6.2. Activating Configuration Changes with Server Re-start

If the LDAP updated server configuration attribute is not one of the general LDAP configuration subentry attributes permitted for dynamic activation, you need to use the **dirxadm ldap restart** operation for the configuration change to become effective. This operation disconnects all LDAP client connections and causes the LDAP server to re-read the directory schema.

For example, suppose the My-Company administrator changes the RPC port number on which the second LDAP server listens for LDAP client requests (the LDAP port number user attribute). To activate this change, he issues the following commands:

```
ldap binding -port 8999
ldap restart
```

so that the new LDAP port number is reflected in the LDAP configuration subentry for the second LDAP server. Note that you must have performed a **dse bind** operation before you can perform another operation in **dirxadm**. (See the **dse bind** operation in the *Administration Reference* for details.)

For the list of LDAP configuration subentry attributes that can be updated dynamically, see the description of the **ldap_cfg_upd** extended operation in the *DirX Directory LDAP Extended Operations*.

### 5.2.7. Using IP Filtering to Control Client Access

The LDAP server configuration subentry contains two attributes that allow DirX Directory administrators to control access to the LDAP server by clients communicating over specific IP addresses:

- LDAP Allow IP List (LAIP or ldapAllowIPList)—specifies IP addresses for which LDAP server access is permitted. By default, all IP addresses are allowed access to the server.
- LDAP Deny IP List (LDIP or ldapDenyIPList)—specifies IP addresses for which LDAP server access is denied. By default, no IP addresses are denied access to the server.

DirX Directory administrators use the **dirxcp obj modify** operation over LDAPv3 to modify the LAIP and LDIP attributes. For example, suppose the My-Company administrator wants to prevent a group of outside consultants who are working on site in the Munich office from accessing the directory. The consultant machines use IP address 192.23.81.56. The My-Company administrator can bind to the contact DSA with **dirxcp** over LDAPv3, and then issue the command:

```
modify /cn=ldapConfiguration,o=My-Company \
       -addattr {ldapDenyIPList=192.23.81.56}
```

The LDAP server uses the following simple strategy to grant or deny access for IPs:

- A client IP must be part of the LAIP (ldapAllowIPList) list.
- A client IP must not be part of the “deny” list.
- Both conditions must be true to grant access, otherwise, access is denied.

As a result, the machines with the IP address 192.23.81.56 will be denied access to the LDAP server. The sections on LAIP and LDIP attributes in the *DirX Directory Syntaxes and Attributes* provide more details.

### 5.2.8. Modifying the Default LDAP Cache Configuration

Recall from the *DirX Directory Introduction*, “LDAP Server Implementation Features” that the DirX Directory LDAP server maintains a configurable and monitorable result cache that can significantly improve performance in cases where LDAP clients send frequent identical search requests. When the LDAP server’s result cache is enabled, the LDAP server searches the cache for a query before it sends the query on to the DSA for processing.

DirX Directory, as delivered, disables the LDAP server’s result cache. DirX Directory administrators can enable the LDAP result cache and configure its operation via attributes in the LDAP server configuration subentry. These attributes are:

- The LDAP Cache attribute, which enables and disables the result cache
- The Maximum Cached Elements attribute, which specifies the maximum number of search requests that can be stored in the result cache
- The Minimum Cache Time and Maximum Cache Time attributes, which specify the duration that an LDAP search result will be stored in the result cache
- The Minimum Cache Entries and Maximum Cache Entries attributes, which specify the range of result entries that an LDAP search result must have in order to be stored in the result cache
- The Cache Table Size attribute, which specifies the size of the internal cache table
- The Cache Update Strategy attribute, which specifies how the LDAP cache is to be updated when the DirX Directory database is modified

Administrators can use DirX Directory Manager or **dirxcp obj modify** operations on the LDAP server’s configuration subentry to enable and configure the LDAP result cache. Changes made to the LDAP cache configuration take effect the next time the LDAP server is started. Administrators can also use the **dirxadm ldap cache** and **ldap restart** operations to enable and disable the cache and apply cache configuration changes to a running LDAP server. The *Administration Reference* provides more information about the **dirxcp** and **dirxadm** tools. The next sections explain how to enable the LDAP cache and provide guidelines for configuring it with the cache configuration attributes.

#### 5.2.8.1. Enabling the Result Cache

An LDAP server’s LDAP Cache attribute is a boolean value that controls whether the caching of LDAP search results is enabled. The DAP abbreviation for this attribute is **LCA**; the attribute does not have an LDAP name.

For example, suppose the My-Company administrator decides to enable the LDAP result cache. He can bind to the DSA over DAP and modify the LDAP server configuration subentry with **dirxcp** as follows:

```
modify /O=My-Company/CN=ldapConfiguration \
       -changeattr {LCA=FALSE}{LCA=TRUE}
```

When the DSA enables the LDAP result cache, it applies default configuration by supplying

default values in its cache configuration attributes. The next sections describe these attributes and their defaults and provide guidelines for their configuration. The *DirX Directory Syntaxes and Attributes* describes the DAP and LDAP syntax for these attributes.

#### 5.2.8.2. Configuring the Cache Size

There is no configurable value for the maximum size of the LDAP result cache. Instead, DirX Directory administrators use the Max Cached Elements and Max Cached Entries attributes to adjust the cache size to the available system memory. The Max Cached Elements attribute limits the number of search results the cache can hold at one time, while the Max Cached Entries attribute limits the number of entries a search result can have in order to be stored in the cache.

An LDAP search result consists of one or more LDAP result messages. Each result message describes one resulting entry. The last result message contains the result code of the search. If the search results in no entry, the LDAP search result consists only of the last result message containing the result code. A cached empty search result (a result that contains just the result code) requires 1KB of memory. The amount of memory required for a typical cached result ( a result with some number of entries) depends upon the size of the retrieved attributes.

By default, the Max Cached Elements value is set to **10,000** elements (a full LDAP cache consumes at least 10MB of memory) and the Max Cached Entries value is set to **0** (empty search results are cached). Administrators can use the following general scenarios to configure the Max Cached Elements and Max Cached Entries attributes to size the cache to their memory requirements:

- Most of the LDAP requests return small results (each result has just a few entries). In this scenario, an administrator should specify a low Max Cached Entries value and a high Max Cached Elements value.
- Most of the LDAP results return large results (each result has many entries). In this scenario, an administrator should specify a high Max Cached Entries value and a low Max Cached Elements value.
- The size of the LDAP results varies between small and large results. In this scenario, an administrator should specify Max Cached Entries and Max Cached Elements values that are lower than the default values

The value given to Max Cached Elements is also related to the size specified for the internal cash table, which is controlled by the Cache Table Size attribute. For best performance, the value of the Cache Table Size attribute should be at least the value of Max Cached Elements divided by 3. The table size must not be lower than **128** and must not be greater than **64,000**. Note that cache performance will decrease if too many results are stored in a table that is too small. The default Cache Table Size is **4096** for a default Max Cached Elements of **10,000**.

#### 5.2.8.3. Configuring the Results that are Cached

The following methods can be used to specify which results are cached:

- The Minimum Cache Entry and Maximum Cache Entry Attributes:

The Minimum Cache Entry and Maximum Cache Entry attributes control which LDAP search results are stored in the result cache. An LDAP search result is only stored in the cache when the number of entries in the result is within the range specified in Minimum and Maximum Cache Entry attributes. By default, the range is from **0** through **2000**.

- The environment variables **DIRX_LDAP_CACHE_BANNED_REQ_ATTR** and **DIRX_LDAP_CACHE_BANNED_BASE_OBJ**:

  The environment variable **DIRX_LDAP_CACHE_BANNED_REQ_ATTR** prevents search results of search requests with specific requested attributes from being cached. The environment variable **DIRX_LDAP_CACHE_BANNED_BASE_OBJ** prevents search results of search requests with specific base objects from being cached.

  For example, the My-Company administrator knows that the entries under the node **ou=state-data,o=my-company** are modified very often. It doesn't make sense to store these entries in the cache because the LDAP server evaluates whether update operations affect the cached results. To exclude these entries from being cached, the administrator specifies the environment variable **DIRX_LDAP_CACHE_BANNED_BASE_OBJ=ou=state-data,o=my-company**

- The environment variable **DIRX_LDAP_CACHE_ALLOWED_USER**:

  The LDAP server stores only search results in the cache if the user's DN is listed in the environment variable **DIRX_LDAP_CACHE_ALLOWED_USER**.

#### 5.2.8.4. Configuring when Results are Removed

The Minimum and Maximum Cache Time attributes combine with other attributes to control how long a cached search result remains the cache. An LDAP search result is removed from the cache when the following conditions are true:

- The maximum number of LDAP search results specified in the Max Cached Elements attribute is exceeded
- The search result has been stored for at least the minimum storage time specified in the Minimum Cache Time attribute
- The maximum storage time specified by the Maximum Cache Time attribute is exceeded
- The search result has the lowest hit rate

By default, a search result remains in the cache for at least **0** seconds and no longer than 12 hours (**43200** seconds).

#### 5.2.8.5. Configuring the Cache Update Strategy

The Cache Update Strategy attribute controls how the result cache is updated when the DirX Directory database is modified by an update operation, so that the cache does not return incorrect results. There are three strategies:

- The "full erase" strategy, in which all results in the cache are removed (0)

- The “DN” strategy, in which all results that may contain the distinguished name of the modified directory entry are removed (1)
- The “attribute” strategy, in which all results that may contain the modified directory entry attribute are removed (2)
- The “DN plus attribute” strategy, in which all results that may contain the distinguished name of the modified entry or the modified attribute are removed (3; this is the default strategy)

Search results without entries will always be removed, and that the DSA will flush the cache before it begins a database restore operation.

### 5.2.9. Adding User Policies to an LDAP Server

The LDAP server configuration subentry contains two attributes that allow DirX Directory administrators to create user-specific and group-specific access policies for clients that connect to an LDAP server:

- LDAP User Policies (ldapUserPolicies or LUSP)—specifies policies that apply to a single user or to a set of users.
- LDAP Group Policies (ldapGroupPolicies or LGRP)—specifies policies that apply to the members of a group.

These attributes extend the LDAP server configuration attributes LDAP Allowed Users (LAUSL), LDAP Allowed Groups (LAGRP), LDAP Denied Users (LDUSL) and LDAP Denied Groups (LDGRP) but are distinct from other LDAP server configuration attributes except for the LDAP Service Controls attributes. For more information about the LDAP User Policies and LDAP Group Policies attributes, see the *DirX Directory Syntaxes and Attributes*.

DirX Directory administrators can use DirX Directory Manager or the **dirxcp obj modify** operation over LDAPv3 to modify the ldapUserPolicies and ldapGroupPolicies attributes and then use the LDAP extended operation **ldap_cfg_upd** from the **dirxextop** command to activate it dynamically. (Dynamic activation is strongly recommended; see the description of the LDAP User Policies attribute in the *DirX Directory Syntaxes and Attributes* for complete details.)

For example, suppose that My-Company decides to create the following LDAP user policies:

- The LDAP search operations for the employees in the Development and Sales departments and the employees of the Frankfurt travel agency are to be limited to returning a maximum of 100 objects per operation. In the ldapUserPolicies attribute, this policy is specified with the USER class value **all** and the SIZELIMIT property:

  ```
  ldapUserPolicies=USER=all:SIZELIMIT=100
  ```

  This setting in SIZELIMIT overrides the value of the LDAP Search Service Controls attribute if its value is lower than the LDAP Search Service Controls attribute. In My-Company’s case, LDAP Search Service Controls attribute is set to return an unlimited

number of objects from a search.

- All the My-Company administrators must bind to the LDAP server using the secure port established for SSL/TLS described in “Enabling SSL/TLS Support”. When this policy is in force, the LDAP server rejects any plain LDAP bind requests made from the administrator. However, the My-Company administrators should have an LDAP search limit that is more flexible than the search return limit of 100 objects specified for all other users. In the ldapUserPolicies attribute, these policies are specified with the WCUSER (wildcard user) class and the TLS_REQUIRED and SIZELIMIT properties:

```
ldapUserPoli-
cies=WCUSER=”*cn=admin*”:TLS_REQUIRED=1:SIZELIMIT=1000000:PRIO=0
```

The TLS_REQUIRED setting is independent of the LDAP SSL Client Authentication Required (LSCAR) attribute, which controls whether client authentication is required over SSL.

By convention, all the My-Company administrators use the common name format **cn=admin**, as described in “Adding the New Administrator Entry”. As a result, the policy defined with **WCUSER="*cn=admin*"** will be applied to all the administrators. However, if an administrator entry uses a different common name format – for example, **cn=main-admin** or **cn=second-administrator** – the entry won’t be covered by the LDAP user policy as defined above.

- Anonymous users are not allowed to perform searches that start at the Development organization subdirectory or to search the default administrator entry and their operations must complete within 10 seconds. In the ldapUserPolicies attribute, this policy is specified with the USER class value **anonymous**, the TIMELIMIT property, the FORB_SB (forbidden search base) property and the FORB_TGT (forbidden target) property:

```
ldapUserPolicies=USER=anonymous:TIMELIMIT=10:FORB_SB=ou=Developmen
t,o=my-company:FORB_TGT=cn=admin,o=my-company
```

- The employees from the Sales department are not allowed to perform searches that start at the Development department. In the ldapUserPolicies attribute, this restriction is specified with the FORB_SB (forbidden search base) property:

```
ldapUserPolicies=SUBR=ou=sales,o=my-
company:FORB_SB=ou=development,o=my-company
```

The FORB_SB policy does not prevent a Sales department employee from searching an entry in the Development department. For example,

> a Sales employee can specify **cn=Josef Smith, ou=development, o=my-company** as the base object for the search operation instead specifying **ou=development, o=my-company**; the search operation that specifies the common name Josef Smith as the search base will succeed, while the search operation that specifies Development as the search base will fail.

The My-Company administrator binds to the contact DSA with **dirxcp** over LDAPv3 and then performs the following steps to enable the policies:

1. Adds the policies attribute to the LDAP configuration subentry:

```
modify cn=ldapConfiguration,o=My-Company -addattr {ldapUserPoli-
cies=USER=all:SIZELIMIT=100;WCUSER="*cn=admin*":TLS_REQUIRED=1:SIZ
ELIMIT=1000000:PRIO=0;USER=anonymous:TIMELIMIT=10:FORB_SB=ou=Devel
opment,o=my-company:FORB_TGT=cn=admin,o=my-
company;SUBUSER=ou=sales,o=my-company:FORB_SB=ou=development,o=my-
company}
```

2. Updates the LDAP configuration dynamically with **dirxextop**:

```
dirxextop -D cn=admin,o=my-company -w dirx -t ldap_cfg_upd
```

Alternatively, the administrator can use DirX Directory Manager to add the policies values and update the configuration dynamically. The advantage of using DirX Directory Manager is that it performs a syntax check on the policies' values.

See the description of the LDAP User Policies attribute in the *DirX Directory Syntaxes and Attributes* for a description of all the properties available for use in the LDAP User Policies attribute.

Administrators can use the LDAP extended operations **ldap_show_policy_rules**, **ldap_show_policy_users** and **ldap_show_single_user_policy_rules** to get information about the current LDAP user and group policies in force, the current status of users registered to the LDAP user policies and the policies that apply to an individual user. See the *DirX Directory LDAP Extended Operations* for details about these operations and examples of their output.

# 6. Creating a Shadow DSA

This chapter describes how to set up a distributed DirX Directory service based only on replication, also called shadowing. Shadowing is a mechanism that automatically copies directory data from one DSA to another over X.500 DISP. DirX Directory administrators can use shadowing to copy any directory tree or subtree (if it is stored in its own database) between two or more DSAs. The DSA that holds the master copy of the shadowed directory information - the **supplier DSA** - automatically copies any updates to that information to all of its slaves the **consumer DSA**s.

Recall from the section on X.500 shadowing in the chapter “Introducing DirX Directory” in the *DirX Directory Introduction* that DirX Directory DSAs can support shadowing with or without a total refresh, also called a **total update**, over DISP. When a shadowing operational binding (SOB) is established between two DSAs and the shadowing process is to be supplier-initiated, the supplier DSA duplicates the shadowed information to the consumer DSA over DISP as soon as the SOB is enabled. Using DISP to perform a total update is an effective means of data duplication for shadowed directories of fewer than one million entries, but it is not the best method for total updates of larger databases. As a result, DirX Directory permits DirX Directory administrators who are shadowing very large directory databases to perform the total update using **dirxbackup** save and restore operations for best performance, provided their shadow configuration replicates the entire DIT.

This chapter describes a sample shadow configuration with the following characteristics:

- The first DSA plays the role of a shadow supplier and is the sample stand-alone DSA established in the chapter "Setting up the DirX Directory Service"
- The second DSA plays the role of an asynchronous shadow consumer and does not master any entries. Its database profile and DSA schema are identical to those of DSA1.
- The complete DIT of DSA1 is shadowed to DSA2.
- The total update of DSA2 is performed via **dirxbackup** save and restore operations.

This configuration is an example of the "floating master" replication configuration discussed earlier in the chapter “Introducing DirX Directory” in the *DirX Directory Introduction*. This chapter describes how to build this configuration and how to use the DirX Directory administration tools to switch the mastership for maintenance purposes and in the event of system failure.

This chapter continues to use the fictitious company "My-Company" to illustrate the planning and administration tasks to set up the sample shadow configuration.

Tcl scripts for building this sample shadow configuration can be found in the directory *install_path***/scripts/ShadowAgreements**. These scripts contain all the commands and procedures discussed in this chapter.

## 6.1. Before You Begin

Before you start to follow the procedures and examples described in this chapter, there is some general information that you should know about how shadowing operates.The

following sections provide details.

### 6.1.1. About Non-Replicated Local Information

The root DSE and the LDAP root subentry contain local information.Shadowing does not propagate modifications to these entries.You must administer the attributes of these entries locally with **dirxadm**, especially the following attributes of the root DSE:

- The DirX Directory Administrators
- The Policy attributes:
- Audit Policy
- DSA Policy
- Local Scope Policy
- Paging Policy
- Search Base Policy
- Time Limit Policy
- User Policy

### 6.1.2. About the Cooperating DSAs Subentry

The cooperating DSAs subentry with the name **/CN=Cooperating-DSAS-Subentry** is a subentry of the root DSE.It contains information about the administered operational bindings with partner DSAs and must be the same for all participating DSAs.

When creating the first shadowing agreement to another DSA, the supplier DSA also creates a shadowing agreement for the cooperating DSA subentry.The DSA administers these agreements automatically.Administrators should never modify or delete these shadowing agreements except in a few erroneous deadlock situations; for example, after a DSA has crashed forever.

(See also the section “Cooperating DSA” in the *DirX Directory Syntaxes and Attributes* for details.)

## 6.2. Planning the Shadow Configuration

My-Company has established a single stand-alone DSA that contains general information about its employees.The amount of employee information is sizable—approximately 32 million entries.The My-Company employees access this directory data through many of the applications in the My-Company enterprise, and this data must be available to these applications at all times.Access to the data must meet the following criteria:

- Maximum availability; that is, if the data is not available on the stand-alone DSA, employees can access the data in another DSA, and vice versa
- The data must be up-to-date and accurate independent of where it is stored.

The solution that meets these criteria is to duplicate all the information contained in My-

Company's stand-alone DSA1 on a second DSA2 in the My-Company enterprise.DSA2 acts as an accessible DSA for load distribution purposes and as an online backup for the directory data that can also be switched to master the directory data, if necessary.The following figure illustrates this solution.

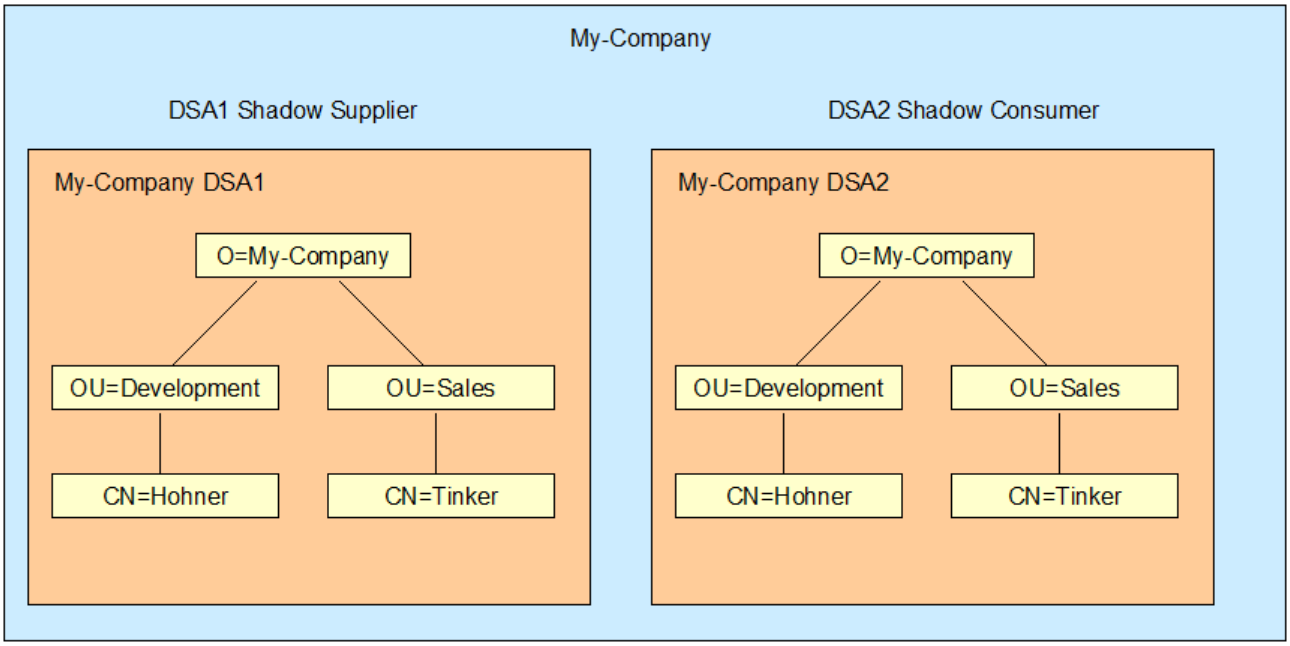



*Figure 28. My-Company Shadow Configuration*

## 6.3. Building the Shadow Configuration

Duplicating data between DSAs requires that the administrators of DSAs to be used in a shadow configuration agree about several points.This understanding is called a **shadowing agreement** and represents the details of what is to be shadowed, when it is to be shadowed, and which role each DSA is to play.

As previously mentioned, the DSA that holds the information to be shadowed is called the **supplier DSA** or the **master DSA**, while the DSA that receives the shadowed information is called the **consumer DSA** or the **shadow DSA**.Supplier and consumer are known as DSA **roles**.The specification of what is to be shadowed is called the **unit of replication**.In this configuration, shadowing is supplier-initiated, where the supplier DSA determines when the shadowing actions take place, and the unit of replication is the complete contents of the DIT.

In My-Company's shadow configuration, a single administrator is responsible for both DSA1 and DSA2, and thus is solely responsible for defining the shadowing agreement between the two DSAs.Once the My-Company administrator has defined the terms of the shadowing agreement, he uses **dirxadm** to create the shadowing agreement on the master DSA (DSA1).This step creates a **Shadowing Operational Binding (SOB)**.The administrator sets up the agreement as "send changes only", which prevents DSA1 from initiating an automatic total update.DSA1 automatically stores the newly created shadowing agreement in the **cooperating DSA table**, which is a subentry that a DSA automatically creates in the DIT during the bootstrap procedure to store all shadowing agreements created after the DSA is up and running.The newly created shadowing agreement is disabled by default to prevent the DSA to send incremental updates while

the shadow consumer is still not set up.

The My-Company administrator next sets up DSA2 with a completely empty database.He then uses **dirxbackup** on DSA1 to create an archive of the DSA's complete DIT and restores this backup with **dirxbackup** onto DSA2.

Now DSA2 is ready to receive the incremental updates from DSA1. The administrator next enables the newly created shadowing agreement on DSA1. This causes DSA1 to propagate the incremental updates saved in the agreement-specific journal file to DSA2. While the shadowing agreement is enabled, any changes made to data within the unit of replication on DSA1 are automatically sent to DSA2 as incremental updates.Incremental updates can be sent immediately (**onChange**) or at bilaterally agreed-upon times (**scheduled**).DSA1 is the master of the shadowing agreement; all changes to the agreement must be made on DSA1, which propagates them to DSA2 as incremental updates.(The incremental update process runs under a special shadowing agreement that the supplier DSA automatically creates for each consumer DSA it supports.)

To establish the shadowing agreement on both DSAs, the My-Company administrator must perform the following tasks:

1. Define the shadowing agreement
2. Create the shadowing agreement on DSA1
3. Save the DSA1 database to an archive
4. Set up DSA2 with a completely empty database
5. Restore the saved DSA1 database on DSA2
6. Enable the shadowing agreement on DSA1

The next sections describe how to perform these tasks.

## 6.3.1. Defining the Shadowing Agreement

First, the My-Company administrator must establish the details of the shadowing agreement between DSA1 and DSA2. These details include:

- Determining the DSA roles and identifying their names and presentation addresses
- Determining the unit of replication
- Determining the shadowing agreement ID
- Determining the consumer DISP kind
- Determining the update mode
- Determining whether the DSA roles can be switched if necessary

#### 6.3.1.1. Determining the DSA Roles

For My-Company, DSA1 is to be the supplier DSA. Its distinguished name is **/CN=DirX-DSA-host1** (environment variable **DIRX_DSA_NAME** on DSA1) and it has the following presentation address (environment variable **DIRX_OWN_PSAP** on DSA1):

| | | |
|---|---|---|
| Transport selector: | | DSA1 |
| Network address: | Internet address: | 123.45.67.89 |
| | port number: | 19999 |

DSA2 is to be the consumer DSA. Its distinguished name is **/CN=DirX-DSA-host2** (environment variable **DIRX_DSA_NAME** on DSA2), and it has the following presentation address (environment variable **DIRX_OWN_PSAP** on DSA2):

| | | |
|---|---|---|
| Transport selector: | | DSA2 |
| Network address: | Internet address: | 198.76.54.32 |
| | port number: | 21111 |

It is important to use real PSAP addresses in shadowing agreements and never to use loopback addresses. The supplier and consumer DSAs use real PSAP addresses in their knowledge information about each other, and chaining will not work without real PSAP address information.

It is also important to use correct DSA names. Incorrect administration of the DSA names will result in a DIRX_DSA_NAME conflict and the DSA will not be able to run.

#### 6.3.1.2. Determining the Unit of Replication

For My-Company, the unit of replication is the entire naming context **/O=My-Company**. All entries within this naming context will be duplicated and all attributes will be duplicated.

#### 6.3.1.3. Determining the Agreement ID

A shadowing agreement requires the presence of an agreement ID that uniquely identifies the shadowing agreement. Since a pair of DSAs can have more than one shadowing agreement, it is up to the administrators of the DSAs to negotiate a unique agreement ID value for each shadowing agreement.

An agreement ID is a sequence of two integers (an identifier value and a version number) separated by a comma. The version number is managed by the DSA. The DSA assigns the value **0** when the agreement is created (with **dirxadm sob create**); each time the agreement is established (with **dirxadm sob establish**), the DSA increments the version number by 1. The My-Company administrator selects the agreement ID **44** to represent this shadowing agreement.

#### 6.3.1.4. Determining the Consumer DISP Kind

A shadowing agreement requires the presence of the consumer DISP kind unless you create a shadowing agreement on a consumer DSA participating on a X.500-compliant shadowing. The DISP kind of a DSA specifies whether the DSA participates on a DirX-Directory-compliant/asynchronous (**CENTRALADMIN**), a DirX-Directory-compliant/synchronous (**SYNCHRONOUS**) or a X.500-compliant shadowing. (See **dirxadm sob create** in the *DirX Directory Administration Reference* for details.)

The My-Company administrator decides that the shadowing is a DirX-Directory-compliant/asynchronous one.

#### 6.3.1.5. Determining the Update Mode

As described earlier, a supplier DSA can send incremental updates immediately ( **onChange**) or at bilaterally agreed-upon times (**scheduled**). A scheduled update strategy is acceptable for shadow DSAs to be used solely for load-balancing of directory information retrieval operations. However, since My-Company's shadow DSA is to be used for high availability of up-to-date directory information, the update strategy for the shadowing agreement must be **onChange**.

#### 6.3.1.6. Enabling Supplier Switching

DirX Directory permits administrators to change the mastership of directory entries between DSAs in a shadow configuration. The My-Company administrator wants to be able to make DSA2 the supplier DSA in the event that DSA1 fails for some reason. To make this situation possible, the administrator must establish a shadow operational binding policy (SOB-policy: the **REPLS** subcomponent of the consumer policy must be **TRUE**.) when he creates the shadowing agreement that allows DSA2 to become a supplier DSA.

### 6.3.2. Creating the Shadowing Agreement on DSA1

To create the shadowing agreement on DSA1, the administrator uses the **dirxadm** command:

**sob create -supplier** *dsa_name* **-supplierpsap** *psap_address*
**-consumer** *dsa_name* **-consumerpsap** *psap_address*
**-consumerkind** *disp_kind*
**-agreementid** *id* **-status** *coop_status* **-agreement** *agreement*

For the agreement on DSA1, the administrator supplies the following values:

- The distinguished name of DSA1 to the **-supplier** option
- The presentation address of DSA1 to the **-supplierpsap** option.
- The distinguished name of DSA2 to the **-consumer** option
- The presentation address of DSA2 to the **-consumerpsap** option
- The keyword **CENTRALADMIN** to the -**consumerkind** option, which specifies a DirX-Directory-compliant/asynchronous shadowing.
- The agreement identifier **44** to the **-agreementid** option
- The keyword **cooperative** to the **-status** option, which activates shadowing upon command completion and means that DSA1 creates the shadowing agreement in the cooperating DSA table.
- An SOB consumer policy (CONS) to the **-pol** option, which specifies that the consumer DSA (DSA2) can become a supplier DSA if necessary (**REPLS=TRUE**).
- The shadowing-subject (SS) operational attribute to the **-agreement** option, which must define:

- The name of the context prefix (CP) to be replicated in the AREA component. This is DSA1's context prefix, which is **/O=My-Company**.
- The entries belonging to this naming context to be replicated in the replication-area (RA) component. The value should be **DEF=TRUE**, which means that the whole naming context must be replicated.
- Whether or not a total update is to be performed in the changes-only (CHANGEO) component. The value should be **CHANGEO=TRUE** to indicate that only changes are to be replicated and that no total update is to be sent.
- The attributes to be replicated in the ATT component. The value should be **DEF=TRUE** to indicate that all attributes are to be replicated.
- The update strategy in the SI subcomponent of the update-mode (UM) component. The value should be **OC=TRUE** to indicate that updates are to be replicated on change.

The following **dirxadm** command creates the shadowing agreement on DSA1:

```
sob create -supplier {/CN=DirX-DSA-host1} \
           -supplierpsap
"TS=DSA1,NA='TCP/IP_IDM!internet=123.45.67.89+port=19999'" \
            -consumer {/CN=DirX-DSA-host2} \
            -consumerpsap
"TS=DSA2,NA='TCP/IP_IDM!internet=198.76.54.32+port=21111'" \
            -consumerkind CENTRALADMIN \
            -agreementid 44         \
            -status cooperative       \
            -pol {CONS={REPLS=TRUE}}  \
            -agreement {SS={AREA={CP={/o=my-company},
                        RA={DEF=TRUE}},ATT={DEF=TRUE}},
                        UM={SI={OC=TRUE}},CHANGEO=TRUE
                       }
```

If this command completes successfully, it displays the following information:

```
AGR={ID=44,VERS=0}
```

When the command is issued, the DSA automatically writes the shadowing agreement information into the Cooperating DSAs subentry **/CN=Cooperating-DSAS-Subentry**. The administrator can use **dirxadm** [**dse**] **show** to display the contents of this subentry. For example:

```
dirxadm show /CN=Cooperating-DSAS-Subentry |
            -allattr -pretty
```

Here is the output of this command: after the agreement has been created:

```
1) /CN=Cooperating-DSAS-Subentry
    DSE-Type                        : CP+SUBENTRY
    Object-Class                    : CDS
                                    : SUBE
    Modification-Time               : 200304301200054Z
    Supplier-Knowledge
      Supplier
        AE-Title                    : /CN=DirX-DSA-host1
        PSAP-Address
          T-Selector                : DSA1
          NSAP-Address              :
              TCP/IP_IDM!internet=123.45.67.89+port=19999
      Agreement-ID
        Identifier                  : 1
        Version                     : 1
    Consumer-Knowledge
      Consumer
        AE-Title                    : /CN=DirX-DSA-host2
        PSAP-Address
          T-Selector                : DSA2
          NSAP-Address              :
              TCP/IP_IDM!internet=198.76.54.32+port=21111
      Agreement-ID
        Identifier                  : 1
        Version                     : 0
     Current-Time-Segment           : -1
      Lower-Time-Segment            : -1
      P-Journal-Offset              : 0
      Maximum-Size                  : 262144
    Cooperating-DSA
      Supplier
        AE-Title                    : /CN=DirX-DSA-host1
        PSAP-Address
          T-Selector                : DSA1
          NSAP-Address              :
              TCP/IP_IDM!internet=123.45.67.89+port=19999
      Consumer
        AE-Title                    : /CN=DirX-DSA-host2
        PSAP-Address
```

```
          T-Selector               : DSA2
          NSAP-Address             :
              TCP/IP_IDM!internet=198.76.54.32+port=21111
        SOB-Agreement
          Shadow-Agreement-Info
            Shadow-Subject
              Area-Specification
                Context-Prefix       : /o=My-Company
                Replication-Area
                  Subtree-Base       : /
              Attribute-Selection
                Default-Value        : TRUE
            Update-Mode
              Supplier-Initiated
                On-Change            : TRUE
          Agreement-ID
            Identifier               : 44
             Version                  : 0
          OprBindMngmnt-Validity
            Validity-From            : 030424121930Z
          Operational-Binding-State  : COOPERATIVE
          Related-Policies
            Consumer
              Replace-Supplier       : TRUE
        SOB-Agreement
          Shadow-Agreement-Info
            Shadow-Subject
              Area-Specification
                Context-Prefix       : /CN=Cooperating-DSAS-Subentry
                Replication-Area
                  Default-Value      : TRUE
              Attribute-Selection
                Default-Value        : TRUE
            Update-Mode
              Supplier-Initiated
                On-Change            : TRUE
            Changes-Only             : TRUE
          Agreement-ID
            Identifier               : 1
            Version                  : 1
          OprBindMngmnt-Validity
```

```
          Validity-From            : 030424121930Z
        Operational-Binding-State  : COOPERATIVE
        Related-Policies
          Supplier
            Supplier-Upd.-Strategy : TOTAL-INCR
          Consumer
            Supplier-Upd.-Strategy : TOTAL-INCR
            Replace-Supplier       : TRUE
```

The administrator can use the **dirxadm sob show** command to display the newly created shadowing agreement on DSA1. For example:

```
sob show -supplier {/CN=DirX-DSA-host1} \
         -consumer {/CN=DirX-DSA-host2} \
         -agreementid 44 -pretty
```

Here is the output of this command:

```
    Validity-Agreement
    Operational-Binding-State  : COOPERATIVE
    OprBindMngmnt-Validity
      Validity-From            : 031107102705Z
      Agreement
        Shadow-Subject
          Area-Specification
            Context-Prefix     : /O=My-Company
           Replication-Area
              Default-Value    : TRUE
          Attribute-Selection
            Default-Value      : TRUE
        Update-Mode
          Supplier-Initiated
            On-Change          : TRUE
        Changes-Only           : TRUE
    Update-Status
      Disabled                 : TRUE
      Update-Status
        Supplier-Update-Status
          Area-Change-State    : CHANGED
    Related-Policies
```

```
      Initiator:
        Recovery-Max.-Retries   : 20
        Recovery-Time-Out       : 1200
        Recovery-Interval       : 240
        Recovery-Divisor        : 16
        Automatic-Total-Update  : FALSE
      Consumer
        Replace-Supplier        : TRUE
```

For your convenience, the **dirxadm sob show** operation displays the values of the shadowing agreement parameters *Changes-Only* and *Initiator* policy even if they were not specified when the shadowing agreement was created. To turn off this display, set the environment variable **DIRX_DSA_OB_SHOW_DISPLAY_DEF_VALUES=0**.

### 6.3.3. Saving the My-Company DIT on DSA1

Now the administrator of DSA1 can use the **dirxbackup** utility to save the directory database into an archive. To save the complete database into an archive file, the administrator uses the **dirxbackup** command:

**dirxbackup -S -v** *archive_name*

The following **dirxbackup** command saves the My-Company DIT to the archive file **My-Company_041103**:

```
dirxbackup -S My-Company_041103
```

In shadow configurations where DSAs run on different operating systems, DirX Directory administrators can use **dirxadm** to save the complete database in an LDIF file, and then **dirxload** to restore it. For example, suppose DSA1 runs on Windows and DSA2 runs on Linux. The My-Company administrator can run the following **dirxadm** command on DSA1 to create an LDIF content file of the complete database:

```
dirxadm create_total_ldif 45
```

The command creates the file in the directory *install_path*/server/ldif* in the name format **0000001112Standalone.***timestamp*, where *timestamp* is a counter increment in milliseconds; for example, **0000001112Standalone.01068022124751**. See **util (dirxadm)** in the *DirX Directory Administration Reference* for further details about this command.

### 6.3.4. Setting up DSA2 as an Empty DSA

The next step in establishing My-Company's shadow configuration is for the administrator to set up DSA2 with an empty database. To carry out this task, the My-Company administrator:

1. Installs DirX Directory from the product CD.
2. Configures and initializes the DBAM database with **dbamconfig** and **dbamboot**; the DBAM device configuration should be identical to the configuration for DSA1.
3. Sets up the environment variable **DIRX_DSA_NAME** on DSA2 with the common name attribute value of DSA2's distinguished name, **/CN=DirX-DSA-host2**. Note that this name must be unique in the replication environment.
4. Sets up the environment variable **DIRX_OWN_PSAP** on DSA2 with DSA2's PSAP address: **TS=DSA2,NA='TCP/IP_IDM!internet=198.76.54.32+port=21200'**.

It is very important to administrate the environment variables **DIRX_DSA_NAME** and **DIRX_OWN_PSAP** correctly. An incorrect value in the **DIRX_DSA_NAME** environment variable will result in a DSA name conflict and the DSA will exit. The DSA is not able to work correctly over the network in DSA to DSA connections with an incorrect value in the **DIRX_OWN_PSAP** environment variable.

### 6.3.5. Restoring the My-Company DIT to DSA2

Now the administrator of DSA2 can use the **dirxbackup** utility to restore My-Company DIT to DSA2's database. To restore a database from an archive file, the administrator uses the **dirxbackup** command:

**dirxbackup -R -v** *archive_name*

The following **dirxbackup** command restores the My-Company DIT to from archive file **My-Company_041103**:

```
dirxbackup -R -v My-Company_041103
```

The shadowing configuration between DSA1 and DSA2 is now complete and operational. From now on, the DSA administrator must be careful not to change the names of DSA1 or DSA2 in the cooperating DSA table subentry or in the **DIRX_DSA_NAME** environment variables of each DSA.

In shadow configurations where DSAs run on different operating systems, DirX Directory administrators can use **dirxadm** to save the complete database in an LDIF file, and then **dirxload** to restore it. For example, suppose DSA1 runs on Windows and DSA2 runs on Linux. The My-Company administrator can run the following **dirxload** command on DSA2 to copy the LDIF content file of the complete database previously created on DSA1 into DSA2:

```
dirxload -r -A -f 0000001112Standalone.01068022124751
```

See the reference pages for **dirxload** in the *DirX Directory Administration Reference* for further details about this command.

Start the DSA.

### 6.3.6. Enabling the Shadowing Agreement on DSA1

After restoring the shadowed data on the consumer DSA, the administrator must explicitly enable the agreement.

To enable a shadowing agreement with **dirxadm**, use the command:

**sob enable -agreementid** *agreement_id* **-supplier** *dsa_name*
**-consumer** *dsa_name*

The DSA1 administrator issues the following **dirxadm** command to enable the shadowing agreement on DSA1 that he previously disabled:

```
sob enable -agreementid 44 \
           -supplier /CN=DirX-DSA-host1 \
           -consumer /CN=DirX-DSA-host2
```

### 6.3.7. Synchronizing Master-Shadow Data

Sometimes a consumer DSA's shadowed directory data can fall out of synchronization with the master directory data on the supplier DSA. The supplier DSA can automatically detect this condition; when it does, it writes an entry into the exception log file and terminates the relevant shadowing agreement. The administrator may also observe this condition. When a shadow DSA's database is out of sync, the administrator should take the following steps:

1. Use the **dirxadm sob terminate** command to terminate the relevant shadowing agreement (unless the supplier DSA has already terminated it). Terminating the agreement erases the incremental updates that have become obsolete.
2. Use the **dirxadm sob establish** command to re-establish but disable the shadowing agreement.
3. Use the **dirxbackup** tool to save the master database and restore it on the consumer DSA.
4. Use the **dirxadm sob enable** command on the supplier DSA to re-enable the shadowing agreement.

### 6.3.8. Replicating the Root Context

If the directory tree contains multiple context prefixes, it can be easier to manage one

shadow agreement starting with the root context instead of creating multiple shadow agreements starting at each context prefix.Creating the shadowing agreement in this case is similar to the procedure previously described, except for the name of the context prefix, which is the root context and is specified with "**/**".

The supplier DSA1 replicates the root DSE and its prefix area plus all context prefixes below the root and all its subsidiary entries.The steps described in sections "Enabling the Shadowing Agreement on DSA1" through "Synchronizing Master-Shadow Data" also apply here.The supplier DSA1 also generates a backup signature in the subentry **/CN=DirXDBVersionSubentry** which will be part of the **dirxbackup** archive or the LDIF dump file(s).

After the archive has been restored on the consumer DSA, the supplier DSA1 evaluates in the enable operation whether the correct archive was restored on the consumer.That is, DSA1 compares the backup signature with the consumer DSA2's signature.For this comparison to succeed, the consumer DSA2 needs to be online.If the backup signature does not match, the enable operation returns an error.

If the described verification procedure does not comply with the operational procedures, you can disable it by setting the environment variable **DIRX_DSA_DISP_TUBM=0**.This setting prevents the generation of the backup signature and comparison between supplier and consumer at enable time.

## 6.4. Using DISP to Perform a Total Update

If your directory database contains less than a million entries, you can use DISP to perform the total update to your shadow DSAs.For example, suppose the My-Company administrator has a much smaller database to shadow.The steps to set up the shadow DSA then are:

1. Determine the shadowing agreement
2. Bootstrap DSA2 (see the chapter "Setting up the DirX Directory Service" for details)
3. Create the shadowing agreement on DSA1 and specify **CHANGEO=FALSE** in the shadowing-subject (SS) attribute.
4. Administer the root DSA on the DSA2; that is, create the DirX Directory Administrators (DADM), DSA Policy (DSAP), and User Policy (USP) attributes with **dirxadm** (see the chapter "Setting up the DirX Directory Service" for details).The DISP protocol does not update the root DSE because it contains the local policies of the DSA.
5. Use the **dirxadm sob establish** command to re-establish but disable the shadowing agreement.
6. Use the **dirxadm sob enable** command on the supplier DSA to re-enable the shadowing agreement.

Once the DSA administrator completes these tasks, the following process occurs:

1. The action of duplicating the data begins.DSA1 initiates a DISP connection to DSA2 and sends the total update for the cooperating DSA subentry, which contains all shadowing agreements.DSA1 then sends the total update as specified by the shadowing

agreement.

2. After the total update is complete, the Shadowing Operational Binding remains active on both sides.During this period any changes made to data within the unit of replication on DSA1 are automatically sent to DSA2 as incremental updates.These updates can be sent immediately (onChange) or at bilaterally agreed-upon times (scheduled).This process occurs as long as the agreement remains active (the same as it is in the **dirxbackup** save and restore scenario).

If the shadowing agreement is disabled on DSA1, incremental updates will no longer be sent to DSA2. Note that if the shadowing agreement is deleted on DSA1, the agreement is automatically deleted from DSA2, but the corresponding entries in DSA2's DIT are not automatically deleted.

If your DSA contain huge entries that cannot be loaded as a whole in memory, for example entries with groups that contain more than 1000 members, you must specify the value **TRUE** for the **Partial Entry Shadowing Policy** attribute.(See section "Partial Entry Shadowing Policy" in the *DirX Directory Syntaxes and Attributes* for details.)

# 6.5. Switching Supplier DSAs in a Shadow Configuration

DirX Directory administrators can switch the mastership of directory entries between DSAs in a shadow configuration.This section describes how to use the DirX Directory administration tools to perform this task:

- On a running system
- When a supplier DSA has failed

The next sections describe these tasks using the My-Company example shadow configuration established in the previous sections.

## 6.5.1. Switching Masters in a Running System

DirX Directory administrators use the following **dirxadm** command to change a supplier DSA (the master of the directory entries in the DIT and the sender of shadowed information) in a shadow configuration:

**sob switch -from** *dsa_name* **-to** *dsa_name*

The **sob switch** command performs the following actions:

- Sets the supplier DSA to a read-only state for the duration of the operation
- Forces the propagation of all outstanding incremental updates for all enabled shadowing agreements
- Updates all the shadowing agreements in the cooperating DSA table to the new supplier-consumer DSA configuration
- Propagates the updated cooperating DSA table to all the consumer DSAs

- Releases the old supplier's read-only state

For example, suppose the My-Company administrator wants to make DSA2 the supplier DSA. He binds to the current supplier DSA, which is DSA1, with **dirxadm** and issues the command:

```
sob switch -from /CN=DirX-DSA-host1
           -to /CN=DirX-DSA-host2
```

This **sob switch** command:

- Sets DSA1 to read-only
- Propagates all outstanding incremental updates for all shadowing agreements to DSA2
- Updates all shadowing agreements in the cooperating DSA table on DSA1 to show DSA2 as the supplier and DSA1 as the consumer
- Propagates the updated cooperating DSA table on DSA1 to DSA2 (and to any other consumer DSAs that are present in the configuration)
- Releases the read-only state on DSA1

### 6.5.2. Switching from a Failed Supplier DSA

When a supplier DSA fails in a shadow configuration, a DirX Directory administrator should remove the DSA from the system and switch mastership to a consumer DSA that contains a complete shadow of the failed master. (See the **dirxadm sob create** operation in the *DirX Directory Administration Reference* how to establish such a stand-by shadow.) The administrator should first ensure that the failed DSA is off-line; he can then issue the **dirxadm sob switch** command on one of the consumer DSAs in the configuration to make it the new supplier DSA.

When the **dirxadm sob switch** command is issued on a consumer DSA, it:

- Updates all the shadowing agreements in the cooperating DSA table to the new supplier-consumer configuration
- Propagates the updated cooperating DSA table to all the shadow DSAs

The command does not force the propagation of incremental updates to the other DSAs in the configuration because the DSA being operated on is a consumer DSA, and thus does not have any incremental updates to propagate. The data of this DSA may differ from the data of the failed master.

For example, suppose the My-Company's administrator has added another consumer DSA **/CN=DirX-DSA-host3** to the My-Company shadow configuration, and its agreement ID is **51**. DSA1 continues to be the supplier DSA; DSA2 and DSA3 are the consumer DSAs. DSA2 has been established as stand-by DSA and therefore contains a complete shadow of the master. Now suppose that the machine that hosts DSA1 fails. To react to this failure, the My-Company administrator decides to switch directory entry mastership. To make this change, the My-Company administrator:

1. Ensures that DSA1 is shut down.
2. Binds to DSA2 with **dirxadm** and issues the **dirxadm** command:

```
sob switch -emergency
```

The **sob switch** command:

- Updates all shadowing agreements in the cooperating DSA table on DSA2 to show DSA2 as the supplier and DSA1 and DSA3 as consumer DSAs.
- Propagates the updated cooperating DSA table on DSA2 to DSA3 (DSA1 is offline).

Keep in mind that the data of DSA2 and DSA3 may differ because DSA3 has not received all updates that DSA2 had received from the old master DSA1.

### 6.5.3. Restoring the Failed Master to the Configuration

The process of reinstating a failed supplier DSA (in the example, DSA1) that has been repaired to a master-shadow configuration consists of adding it as a consumer DSA and then switching directory mastership to it. Recall from the section “Building the Shadow Configuration” that DirX Directory administrator has configured all agreements to perform a total update by media to synchronize initially the shadowed directory data.

A DirX Directory administrator can use the following procedure to return it to the running system:

1. On the current supplier DSA (in the example, DSA2), terminate and then re-establish the shadowing agreements between the supplier DSA (DSA2) and the DSA (DSA1) to be re-entered into the configuration. The agreement is automatically disabled.
2. Use the **dirxbackup** save (on DSA2) and restore operations to perform a total update on the old supplier DSA (DSA1).
3. Start the old supplier DSA (DSA1), which now becomes a consumer DSA in the configuration.
4. Enable the corresponding shadowing agreements on the current supplier DSA (DSA2); this action propagates any updates that occurred during the save and restore operations to the consumer DSAs (DSA1 and DSA3).
5. (Optional) Use the **dirxadm sob switch** command on the current supplier DSA (DSA2) to change the master-shadow roles. After this operation has finished, the repaired old master DSA (DSA1) is again the supplier DSA (and DSA2 a consumer DSA).

For example, to restore DSA1 to the My-Company shadow configuration as the supplier DSA, the administrator:

1. Binds to DSA2 with **dirxadm** and uses the **sob terminate** command to terminate all shadowing agreements between DSA2 and DSA1. In the example, these are the shadowing agreement with the identifier **44** and the agreement that propagates the modifications to the cooperating DSA table to DSA1. The administrator must only

terminate the shadowing agreement with the id 44 because the DSA updates all agreements propagating the cooperating DSA table automatically. (All agreements that the DSA manages automatically have got an identifier greater or equal the value 10000. Never modify such agreements manually.)

```
sob terminate -agreementID 44 \
              -supplier /CN=DirX-DSA-host2 \
              -consumer /CN=DirX-DSA-host1
```

This action deletes any accumulated incremental updates for the target consumer, which are obsolete because a total update (via **dirxbackup** save and restore) will be performed.

2. Uses the **dirxadm sob establish** command to re-establish as disabled shadowing agreement **44**:

```
sob establish -agreementID 44 \
              -supplier /CN=DirX-DSA-host2
              -consumer /CN=DirX-DSA-host1
```

3. Uses the **dirxbackup -S** command on DSA2 to save the My-Company DIT to an archive:

```
dirxbackup -S -v My-Company_073103
```

4. Uses the **dirxbackup -R** command on DSA1 to restore the DSA2 database archive to DSA1:

```
dirxbackup -R -v My-Company_073103
```

5. Starts the DirX Directory Service on DSA1. The DSA will start as a consumer DSA.
6. Uses the **dirxadm sob enable** command on DSA2 to enable the shadowing agreement **44** and propagate any updates made since the save operation on DSA2 to DSA1:

```
sob enable -agreementid 44 \
           -supplier /CN=DirX-DSA-host2
           -consumer /CN=DirX-DSA-host1
```

7. Uses the **dirxadm sob switch** command on DSA2 to make DSA1 the supplier DSA:

```
sob switch -from /CN=DirX-DSA-host2
```

```
            -to /CN=DirX-DSA-host1
```

Recall from the section titled “Switching from a Failed Supplier DSA” that the administrator has introduced DSA3 as a consumer DSA.DSA3 receives updates from shadowing agreement **51**.After performing the **dirxadm sob switch** command, DSA1 is the supplier for shadowing agreement **44** (DSA1 to DSA2) and **51** (DSA1 to DSA3).

# 6.6. Password Policies in a Shadow Configuration

This section provides information about password policies in a shadowing configuration.

## 6.6.1. Maintaining Local vs. Global Password Policy States

Password policy features such as account locking or password aging are supported by directory operational attributes that store password policy state information at the respective user entry.Password policy state changes occur as a result of modify operations that deal with the User Password attribute or as a result of successful or unsuccessful bind operations.While modify operation state changes are replicated by normal protocol, bind operation state changes are not.Examples of modifications performed in the context of the bind operation include removing outdated Password Failure Time values after a bind or adding the Password Account Locked Time timestamp after a repeated failing bind.

### 6.6.1.1. Maintaining Local Password Policy States

By default, DirX Directory maintains password policy-related operational attributes with a local scope for modifications that occur during the bind operation.

DirX Directory’s implementation of the simple bind operation in the DSA strictly follows the X.500 approach. The bind operation has a local scope; that is, the user entry with its User Password attribute must be contained in the DSA’s local database regardless of whether it is master or shadow information. A bind is never chained to another DSA. Modifications that are applied to a user entry as a result of a bind operation are always performed only in the local database, including modifications that occur as a result of password policy decisions that potentially affect one or more password policy-related operational attributes of the entry.

As a result, the view of the account locking and password aging features may differ from DSA to DSA in a replicated scenario. For example:

- A user‘s account may be locked on DSA1, but failing binds are still allowed on DSA2.
- A user‘s password may have expired on DSA1, but grace binds are still available on DSA2.

While this is an acceptable situation in most application scenarios, the fact remains that a condition like “lock-account-after-3-consecutive-failed-logins-within-5-minutes” is not enforced when you consider the whole system with all possible bind target DSAs.

Maintenance of local password policy states is the only option supported by DirX Directory V8.5 and older. In DirX Directory V8.6, maintenance of local password policy states is the default and is activated by setting the environment variable **DIRX_GLOBAL_PPO_STATES**

to **0** or by not setting this environment variable at all.

#### 6.6.1.2. Maintaining Global Password Policy States

To create a consistent view of the account locking and password aging features in a replication scenario, you can enable the maintenance of global password policy states by setting the environment variable **DIRX_GLOBAL_PPO_STATES** to **1** in the DSA environment; for example, in *install_path***/conf/dirxenv.ini**.

When **DIRX_GLOBAL_PPO_STATES** is set to **1**, each modification of an entry's password policy-related operational attributes that occurs during the bind operation is performed by an implicit DAP modify operation. Although this DAP modify operation is not received via protocol from an external client, it results in the respective replication protocols:

- On a shadow supplier DSA, the modification is replicated via the DISP protocol to all shadow consumer DSAs with the entry in the replication area according to the rules of the shadow agreement.
- On a shadow consumer DSA, the modification is chained via the DSP protocol to the shadow supplier DSA.

Because standard replication is used, the distribution of the modified operational attributes is subject to replication delay; that is, the propagation of a password policy state change typically occurs on a shadow consumer DSA a couple of milliseconds after the state change becomes effective on the shadow supplier DSA, depending on the distance and quality of the network connection.

You can avoid this replication delay by configuring synchronous shadow agreements. In this case, the state changes are performed completely synchronously; that is, the bind operation returns to the client application after the necessary state changes have been performed on all participating DSAs.

You can also use scheduled agreements to couple shadow supplier and shadow consumer DSAs; in this case, the state changes are synchronized according to the specified schedule. In general, scheduled agreements are not suitable for providing a consistent view of account locking and password aging.

An entry's state attributes are only kept consistent if the shadow agreement is enabled.

If replication is temporarily disrupted; that is, the shadowing agreement is in the error-recovery or disabled state, the following rules apply:

- State changes that occur at the shadow supplier DSA are buffered in the shadowing journal of the supplier DSA and replayed as soon as the shadowing process is re-enabled.
- No state changes are invoked by bind operations that occur at the shadow consumer DSA during the disabled state. Bind operations to a temporarily disconnected shadow consumer DSA work based on the state resulting from the last successful replication. The password policy state is synchronized with the one on the shadow supplier DSA after the shadowing process is re-enabled.

If the shadowing agreement is terminated and set to the NON-COOPERATIVE state, the password policy state changes that occur at the former shadow consumer DSA are maintained locally as for any other stand-alone DSA.

The following password policy-related operational attributes can be changed by implicit modify operations:

- Password Failure Time
- Password Account Locked Time
- Password Expiration Warned Time
- Password Grace Use Time

Shadow agreements must not exclude these operational attributes in their specification.

The following password policy-related operational attributes are also modified during the bind operation but are not used by the DSA for any password policy decisions and are not kept globally consistent. These attributes are only maintained on the local bind DSA independent from the setting of **DIRX_GLOBAL_PPO_STATES**:

- Password Number Of Failed Logins (local)
- Password Last Login Time (local)

You do not need to add an explicit grant in the access control settings for modifications to the password policy-related operational attributes. The DSA automatically recognizes the implicit modifications and access to these attributes is automatically granted to the requestor of the operation (that is, to the DSA) regardless of the settings specified in the Access Control subentries for users.

The implicit DAP modification representing the state change and the resulting DSP chaining and/or DISP replication operations are shown in the DSA audit like all other operations.

Do not enable the maintenance of global password policy states unless all the DSAs participating in your shadowing network are DirX Directory V8.6 or newer. The environment variable **DIRX_GLOBAL_PPO_STATES** should be set to **1** on all participating DSAs.

### 6.6.2. Unlocking an Account in a Shadow Configuration

If a user's account has been permanently locked or the user's account needs to be unlocked before the lock duration timer is over, the administrator must unlock the associated account. To make the unlock operation effective on all DSAs (regardless of whether it is hosting master or shadow information and regardless of the local or global password policy state setting), the administrator should perform the following steps:

1. Perform an LDAP bind with **dirxcp** to the master DSA of the entry to be unlocked.
2. Perform the following **obj modify** operation:

```
modify dn -add pwdAccountLockedTime=200001010000000Z
          -remove pwdAccountLockedTime
          -add pwdFailureTime=200001010000000Z
          -remove pwdFailureTime
         [-replace userPassword=password]
```

where *dn* is the distinguished name of the entry to be unlocked and *password* is the new password if the administrator wants to re-set the user's password.

If the user's entry is inside the replicated area, the entry is unlocked on all participating DSAs after shadowing has propagated the update.

### 6.6.3. Handling the Password Must Change Condition in a Shadow Configuration

If the Password Policy subentry's Password Must Change attribute (pwdMustChange, PPMUCH) is set to TRUE and an administrator has reset a user entry's password, all operations are prohibited after the bind operation except for an operation to change the user's password.All prohibited operations are rejected with the error code "unwilling to perform" and the error message "Password Modification is the only operation allowed due to Password Policy".

After binding to a shadow DSA, a user may invoke a modify operation that deals with attributes that are distinct from the user password.Because this modify operation is chained to the master DSA, this operation is not rejected with the "unwilling to perform" error code and error message.The master DSA performs the operation.

This action applies only to modify operations.

Proper access control decisions are not affected.

## 6.7. Shadowing in a Heterogeneous Network

In DirX Directory, shadowing agreements are administered on the master DSA of the cooperating DSA table.This DSA sends all modifications of the cooperating DSA table to all other DSAs participating in the network.That is, in DirX Directory, shadowing agreements are administered once and distributed by shadowing itself.

X.500-compliant shadowing, however, does not support this central administration of shadowing agreements.It only supports a local administration of shadowing agreements, where administrators must administer an agreement on all DSAs taking part in this agreement.

In DirX Directory, it is possible to administer a DSA for X.500-compliant or for DirX-Directory-compliant central shadowing administration through the *disp_kind* value of the **-supplierkind** and the **-consumerkind** options of the **dirxadm sob create** operation.(See the **dirxadm sob create** operation in the *DirX Directory Administration Reference* for

details.) Once a DSA is administered as X.500- or DirX-Directory-compliant in the cooperating DSA table, administrators cannot specify the *disp_kind* value in new agreements for this DSA.Keep in mind that you never specify the *disp_kind* value for the DSA to which you are bound.Note that the *disp_kind* value also controls whether a DirX-Directory-compliant DSA performs asynchronous or synchronous shadowing; this topic is discussed in more detail in the chapter “Creating a Synchronous Shadow DSA”.

If a DSA—for example a non-DirX Directory DSA—is administered as X.500 compliant, the shadowing agreement must also be specified locally on that DSA.This DSA can act as a supplier or a consumer of a shadowing agreement.It is recommended to administer the shadowing agreements first on the consumer DSA.

Recall from the sections above how the My-Company administrators have set up shadowing.Now the My-Company management has bought a small company enhancing the product portfolio of My-Company.Like My-Company, the new company runs its own directory.Unfortunately, this directory is not DirX Directory, but it is X.500 compliant.

DSA 1 masters the My-Company data and the cooperating DSA table. Shadowing between My-Company’s DSA 1 and DSA2 has already been set up at the beginning of this chapter.

So now the administrators decide that they want to set up shadowing agreements where

- The My-Company DSA 1 additionally sends the My-Company data via DISP to the new company part DSA. This DSA is referred to as DSA 3. DSA 1 is the supplier and DSA 3 is the consumer in this shadowing agreement.
- DSA 3 sends the new company data via DISP to DSA 1 and DSA 2. In these shadowing agreements, DSA 3 acts as the supplier and DSA 1 and DSA 2 as consumers.

The necessary shadowing agreements for the old My-Company network part must be administered all on DSA 1, the master of the cooperating DSA table. DSA 1 then distributes the agreements and any subsequent modifications to this agreement to DSA 2.

All necessary shadowing agreements for the new My-Company part must be administered locally on DSA 3 because DSA 3 is a non-DirX Directory DSA and cannot participate in the central cooperating DSA table administration.

The following sections provide information on how to set up the shadowing configuration on the participating DSAs.

### 6.7.1. Setting up Shadowing for Distributing My-Company Data

As mentioned above, My-Company DSA 1 masters the My-Company data. It already distributes the My-Company data via DISP to DSA 2. Now it should also distribute this data to DSA 3. For this agreement, DSA 1 is the supplier DSA and DSA 3 is the consumer DSA.

The administrators of DSA 1 and DSA 3 take the following steps to set up the shadowing configuration:

1. First, the administrator of DSA 3 creates a consumer shadowing agreement locally to signal that DSA 3 is ready to receive first the total update and then subsequent modifications.

2. Next, the My-Company administrator creates the supplier shadowing agreement for DSA 1 with the X.500-compliant consumer DSA 3 using **dirxadm** on the master DSA of the cooperating DSA table (also DSA 1):

```
sob create -supplier {/CN=DirX-DSA-host1} \
           -consumer {/CN=DSA-host3} \
           -consumerkind X500
           -consumerpsap
"TS=DSA3,NA='TCP/IP!internet=198.76.54.35+port=21111'" \
            -agreementid 45         \
            -status cooperative       \
            -agreement {SS={AREA={CP={/o=my-company},
                        RA={DEF=TRUE}},ATT={DEF=TRUE}},
                        UM={SI={OC=TRUE}}
                       }
```

DSA 1 immediately sends a total update to DSA 3 because DSA 1 is also the supplier of the shadow information.

### 6.7.2. Setting up Shadowing for Distributing New Company Part Data

As mentioned above, DSA 3 should now send the new company part data via DISP to the My-Company DSAs DSA 1 and DSA 2. DSA 3 acts as the supplier and DSA 1 and DSA 2 as consumer DSAs.

The administrators perform the following steps to set up the shadow configuration:

1. On DSA 1, the master of the cooperating DSA table, the My-Company administrator uses **dirxadm** to create the following consumer shadowing agreements for DSA 1 and DSA 2 with the X.500 compliant supplier DSA 3:

   i. Consumer shadowing agreement for DSA 1:

```
sob create -supplier {/CN=DSA-host3} \
           -supplierpsap
"TS=DSA3,NA='TCP/IP!internet=198.76.54.35+port=21111'" \
            -supplierkind X500 \
            -consumer {/CN=DirX-DSA-host1} \
            -agreementid 46         \
            -status cooperative       \
            -agreement {SS={AREA={CP={/o=my-NewCompany},
                        RA={DEF=TRUE}},ATT={DEF=TRUE}},
                        UM={SI={OC=TRUE}}
```

```
                                }
```

DSA 1 is ready immediately to receive a total update and subsequent modifications from DSA 3.

You must provide the **-supplierkind** option here in order to instruct DSA1 to accept DISP updates from DSA3.

ii. Consumer shadowing agreement for DSA 2:

```
sob create -supplier {/CN=DSA-host3} \
           -supplierkind X500 \
           -consumer {/CN=DirX-DSA-host2} \
           -agreementid 47         \
           -status cooperative       \
           -agreement {SS={AREA={CP={/o=my-NewCompany},
                       RA={DEF=TRUE}},ATT={DEF=TRUE}},
                       UM={SI={OC=TRUE}}
                      }
```

DSA 2 is ready to receive a total update and subsequent modifications from DSA 3 after DSA 1 has distributed the cooperating DSA table modification to DSA 2.

It is not necessary to specify the supplier and the consumer PSAP addresses because they have already been specified in the other shadowing agreements.

2. Next, the administrator of DSA 3 creates two supplier shadowing agreements locally to achieve that DSA 3 sends first the total update and then subsequent modifications to DSA 1 and DSA 2.

# 6.8. Managing Unique Constraint Checks in a Shadow Configuration

Unique constraints checks of attribute values are usually performed on the supplier DSA.However, a switchable consumer DSA (where the **REPLS** component of the consumer policy is **TRUE**) also needs to perform these checks because in the case of a **dirxadm sob switch** operation, DirX Directory must prove uniqueness without interruption.It is the administrator's responsibility to configure a consistent unique constraint configuration across the relevant DSAs: every unique constraint setting must match on the supplier DSA and all switchable consumer DSAs.

Configuration always begins on the supplier DSA with the following command for each required attribute type, one after the other:

**dirxadm db attrconfig** *attribute_type* **-index INITIAL UNIQUE**

If post indexing detects non-unique attribute values, the operation returns an error and the **UNIQUE** flag is not set while the **INITIAL** index remains set.Non-unique attribute values are reported in the DSA's logging file.Usually modifications of the involved entries are performed manually.

Repeat the procedure until the **UNIQUE** flag is set for all required attribute types.Use the **dirxadm db show** command to control the attribute index characteristics.

Once the supplier DSA's configuration is stable, proceed with the switchable consumer DSA's unique constraint configuration. During this time, the supplier DSA must be online because the switchable consumer DSA looks at the supplier DSA's database configuration subentry to evaluate the **UNIQUE** flag. Note that sufficient access rights must be established; that is, read permission must be granted for attributes vendorName, vendorVersion and attributeIndex.

We recommend that you implement a consistent unique constraint configuration before applying a switch operation. For a non-emergency switch, the new designated supplier DSA's unique constraint configuration must match the old supplier DSA's configuration. If not, the switch operation is rejected.

For an emergency switch, checking the unique constraint configuration is not performed due to the nature of the situation. The administrator must implement a consistent configuration if he or she has not already done so. Note that a consumer DSA checks its configuration at startup against the supplier DSA and removes unique constraint settings if they are not set on the supplier DSA.

If unique constraint checks are to be deactivated, start with the switchable consumer DSAs first. Perform the following command for all attribute types required:

**dirxadm db attrconfig** *attribute_type* **-index NO-UNIQUE**

Finally, perform this step on the supplier DSA. At this time, all switchable consumer DSAs must be online since removal of the **UNIQUE** flag is proven. If this flag is not removed on any switchable consumer DSA or a consumer DSA is not online, the operation returns an error. One exception results in switchable consumer DSAs that are off-line and their agreements are disabled. They are excluded from the check and won't reject the operation.

Non-switchable consumer DSAs (the **REPLS** component of the consumer policy is **FALSE**) should never be configured for unique constraint checking since the supplier DSA already performs this task.

Due to the lack of functionality, non-compatible DirX Directory V8.2 DSAs—DirX Directory DSAs Version 8.1 and older or non-DirX Directory DSAs—should perform the role of non-switchable consumer DSAs if unique constraint checking is planned to be applied.

# 7. Distributing the DIT across Multiple DSAs

This chapter describes how to set up a true distributed DIT, where different DSAs each hold pieces of an organization's DIT and communicate with each other to present a "seamless" view of the DIT.

The chapter describes a sample distributed configuration with the following characteristics:

- The superior DSA is the sample stand-alone DSA established in the chapter "Setting up the DirX Directory Service".
- The subordinate DSA has the following characteristics:
  - Its administrative point is at the organization-unit level and is an autonomous administrative point
  - The top level of its part of the DIT is at the organization-unit level

This chapter continues to use the fictitious company "My-Company" to illustrate the planning decisions and administrative tasks necessary to set up the sample distributed configuration.

## 7.1. Planning the Distributed Configuration

My-Company has established a stand-alone DSA that contains general information about the employees in its sales and development organizations and shadows this information on a DSA in Hamburg.My-Company now plans to extend its DIT with information about employees in its manufacturing organization.

The first DSA, called DSA1, is located in Munich at the company's headquarters.This DSA stores the employee information about the company's sales and development organizations.My-Company plans to set up a third DSA, called DSA3, at the manufacturing plant in Durach to store the information about the manufacturing employees.The following figure illustrates this configuration.

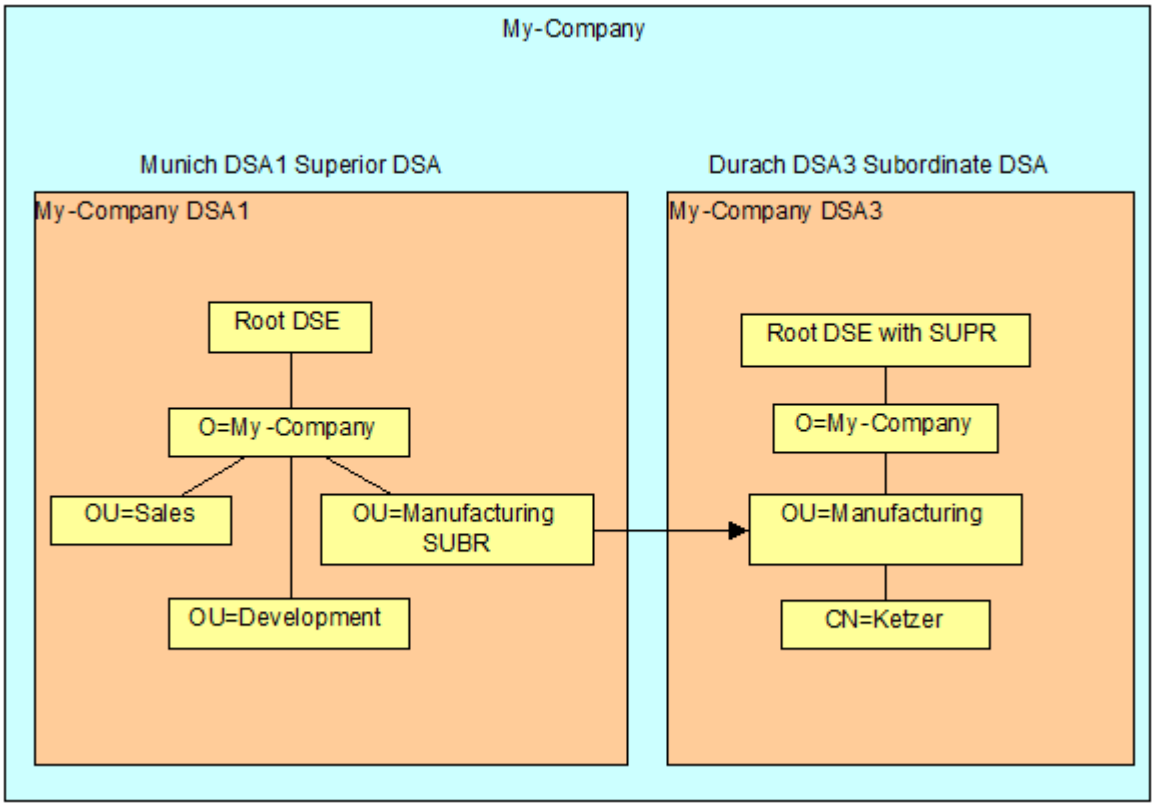



*Figure 29. My-Company Distributed DIT*

The Munich DSA1 is the **superior DSA** and the Durach DSA3 is the **subordinate DSA**.

# 7.2. Building the Distributed Configuration

Building My-Company's distributed configuration consists of two steps:

1. Planning and building the new subordinate DSA
2. Establishing communications between the two DSAs so that they each have information about the other and the piece of the DIT each holds

The next sections describe these steps.

## 7.2.1. Planning the Subordinate DSA

Recall that when setting up their first DSA, My-Company needed to make decisions in two areas before building the DSA:

- Defining what the DIT will be
- Defining the framework for DSA administration (DSA name, a default administrator, access rights, DUA name and presentation address)

My-Company must evaluate these same issues before it can build its new DSA.

#### 7.2.1.1. Determining the Subordinate DSA's DIT

In order to define the DIT, My-Company must define:

- The structure of this part of the DIT

  The DIT held by DSA3 is an extension of the DIT held by DSA1. It consists of one organizational unit - Manufacturing - with employees underneath this level. Manufacturing is represented as an object of object class organizational-unit, and the employees are represented as entries of object class organizational-person.

- The DIT names in the upper part of this DIT

  The DIT on DSA3 begins at the organizational-unit level. Consequently, the topmost entry is **/O=My-Company/OU=Manufacturing**.

- The attribute types that will be used to store employee information

  The DIT held by DSA3 needs to store information about the Manufacturing organizational unit and the manufacturing employees. However, this DIT does not need to store the general information about My-Company. The information about the Manufacturing organizational unit and its employees has the same storage requirements as the information about the Sales and Development organizational-units and their employees. Consequently, this DIT will use the same attribute types for organizational-unit and organization-person object classes as does DSA1.

#### 7.2.1.2. Determining the Administrative Framework

The next planning step for My-Company is to establish the administrative framework in which DSA3's administration will be performed. This planning process consists of the following tasks:

- Defining a DSA name and presentation address

  The name for DSA3 is **/CN=DirX-DSA-host3** (environment variable **DIRX_DSA_NAME** on DSA3). Its presentation address (environment variable **DIRX_OWN_PSAP** on DSA3) is:

| **Transport selector:** | | **DSA3** |
|---|---|---|
| Network address: | Internet address: | 111.22.33.44 |
| | port number: | 23333 |

- Defining the location of the first administrative point

  The topmost entry held by DSA3 is **/O=My-Company/OU=Manufacturing**; this is the first administrative point and is also DSA3's context prefix. DSA3's first administrative point is an autonomous administrative point. Because autonomous administrative points are exempt from the policies of superior administrative areas and access control policies defined at the **/O=My-Company** level do not apply within the **OU=Manufacturing** and must be redefined during DSA3's initialization.

  All superior name parts of the context prefix of the first administrative point must be introduced to the DSA so that it can build complete distinguished names for all of its entries. In this case, **/O=My-Company** is a superior of DSA3's context prefix, which is **OU=Manufacturing**. However, DSA1 masters the entry **/O=My-Company**, and so DSA3

cannot hold another master entry of **/O=My-Company**, because there is always only one master of one entry. Consequently, a "glue" must be established for **/O=My-Company**. A glue provides the name of a superior of the context prefix; it is not an entry, and is not accessible through the protocol.

- Defining the default administrator

  My-Company determines that the name of the default administrator entry for DSA3 is **/O=My-Company/OU=Manufacturing/CN=admin**.

- Defining the access rights

  My-Company decides that the default administrator for DSA3 will have the same access rights on DSA3 as the default administrator **/O=My-Company/CN=admin** has on DSA1. In addition, the DSA1 default administrator will have the same rights as the DSA3 default administrator **/O=My-Company/OU=Manufacturing/CN=admin** on DSA3.

  Users on DSA3 will have the same access rights as the users on DSA1.

- Defining a DUA Address

  In order for DUAs, especially **dirxcp**, to bind to a DSA, a DUA must be assigned a presentation address. If the DUA runs on the same machine as the DSA, the DUA's transport selector and the port number must be different from the DSA's values.

  My-Company uses a local client with the DSA with the following presentation address:

| **Transport selector:** | | **DUA3** |
|---|---|---|
| Network address: | Internet address: | 111.22.33.44 |
| | port number: | 23456 |

With these planning decisions made, the administrator of DSA3 can now build the subordinate DSA.

## 7.2.2. Building the Subordinate DSA

The next step is to build DSA3 as a stand-alone DSA. To perform this task, the administrator follows the procedures for setting up a stand-alone DSA, which are:

1. Bootstrapping the DSA
2. Initializing the DSA
3. Populating the tree

#### 7.2.2.1. Bootstrapping the DSA

To perform this task, the administrator of DSA3 follows the same procedures as described in the section "Bootstrapping the DSA" in the chapter "Setting up the DirX Directory Service". The only differences for DSA3 are that:

- The glue established for DSA3 is **/O=My-Company**
- The DSA name for DSA3 is **/CN=DirX-DSA-host3**
- The administrative point and default administrator are one level lower than for DSA1:
  - The first administration point is **/O=My-Company/OU=Manufacturing**
  - The default administrator is **/O=My-Company/OU=Manufacturing/CN=admin**
- The default administrator of DSA1 **/O=My-Company/CN=admin** receives the same access rights within the subentry ACI as the DSA3 default administrator; that is, two distinguished names are given in the SACI attribute
- The DirX Directory Administrators attribute for DSA3 contains the distinguished name of the administrator of DSA3

##### 7.2.2.1.1. Establishing the DSA Name and Presentation Address in DSA3

Recall from the chapter "Setting up the DirX Directory Service" that a DSA will not bootstrap completely until it knows its own name and presentation address. The environment variables **DIRX_DSA_NAME** and **DIRX_OWN_PSAP** specify a DSA's name and presentation address. The values for DSA3 are:

**DIRX_DSA_NAME="/CN=DirX-DSA-host3"**

**DIRX_OWN_PSAP="TS=DSA3,NA='TCP/IP_IDM!internet=111.22.33.44+port=23333'"**

##### 7.2.2.1.2. Creating the Glues for DSA3

To create a glue, use the command

[**dse**] **create** *distinguished_name* **-attribute DSET=GLUE**

Supply the distinguished name of the glue in the *distinguished_name* argument. The presence of the DSE-type (DSET) operational attribute indicates that the entry is a glue; do not specify any other attributes.

The following **dirxadm** command creates the glue **/O=My-Company** for DSA3:

```
create /O=My-Company -attribute DSET=GLUE
```

##### 7.2.2.1.3. Creating the First Administrative Point for DSA3

The following **dirxadm** command creates the first administrative point for DSA3:

```
create /O=My-Company/OU=Manufacturing -attr \
      {OCL=TOP;OU} \
      {DSET=ENTRY+ADM_POINT+CP} \
      {AR=AA;ACSA;SASA}  \
      {ACS=SACS} \
```

```
      {SACI={ID=admin: enable Handling of Subentries,
             PR=254,
             AL={BL={L=SIMPLE}},
        UF={UC={N={DN={/O=My-Company/OU=Manufacturing/CN=admin}};
                  {DN={/O=My-Company/CN=admin}}},
                UP={PI={E=TRUE},
                    GAD=grantRead+grantBrowse+\
                        grantDiscloseOnError+\
                        grantReturnDN+\
                        grantAdd+grantRemove+grantModify};
                   {PI={AT=SS;OC;AT;DSR;NF;
                        DCR;MR;MRU;CRN;CRT;MN;MT;PACI;EACI,
                        AAV=SS;OC;AT;DSR;NF;
                        DCR;MR;MRU;CRN;CRT;MN;MT;PACI;EACI,
                        AUATV=TRUE},
                    GAD=grantDiscloseOnError+\
                        grantRead+grantCompare+\
                        grantFilterMatch+\
                        grantAdd+grantRemove} } } } \
      {CRN={/O=My-Company/OU=Manufacturing/CN=admin} } \
      {PA={PA=My-Company,PA=High Street 123,\
           PA=D-23456 Durach,PA=Germany}} \
      {TN=+49 68 345 67 890} \
      {DSC=My-Company}
```

#### 7.2.2.1.4. Creating the Default Administrator Entry

For DSA3, an administrator with the distinguished name **/O=My-Company/OU=Manufacturing/CN=admin** must be created. The entry is of object class organizational-person, and a user password will be used during authentication. The following **dirxadm** command creates the default administrator entry for DSA3:

```
   create  /O=My-Company/OU=Manufacturing/CN=admin \
         -attr   {OCL=TOP;PER;ORP} \
                 {DSET=ENTRY} \
                 {SN=admin} \
                 {DSC=Default Administrator} \
                 {UP=dirx} \
                 {CRN={/O=My-Company/OU=Manufacturing/CN=admin}}
```

#### 7.2.2.1.5. Controlling Access to dirxadm

To add the DSA3 administrator distinguished name to the DirX Directory Administrators attribute (DADM attribute) for DSA3, the DSA3 administrator issues the **dirxadm** command:

```
modify / -addattr \
         DADM={/O=My-Company/OU=Manufacturing/CN=admin}
```

The administrator must now use **dirxadm** to bind to DSA3 with simple authentication and the password **dirx**.

### 7.2.2.2. Initializing the DSA

To perform this task, the administrator of DSA3 follows the same procedures as described in the section “Initializing the DSA” in the chapter “Setting up the DirX Directory Service”. The only differences for DSA3 are that:

- The default DSA specified in the client configuration file **dirxcl.cfg** is DSA3
- The access control subentry is one level lower:
  - **/O=My-Company/OU=Manufacturing/CN=AccessControl-Subentry**
- In the **dirxcp bind** operation, the local administrator is **/O=My-Company/OU=Manufacturing/CN=admin**
- The PACI attribute in the access-control-subentry contains two distinguished names, giving the same rights as DSA3’s default administrator to the DSA1 administrator **/O=My-Company/CN=admin**

#### 7.2.2.2.1. Establishing the DSA Name and Presentation Address in the Client

Recall from the chapter “Setting up the DirX Directory Service” that the name of the DSA and its presentation address must be made available to the **dirxcp** program. The DUA name and presentation address must also be established.

The DUA presentation address and DSA name and presentation address are provided to the client in the file **dirxcl.cfg**. The DSA3 administrator modifies this file to contain the presentation address of the DUA and the name and presentation address of My-Company-DSA3 as the default DSA:

```
# File: dirxcl.cfg
# first non-comment line is client address
self TS=DUA3,NA='TCP/IP_IDM!internet=111.22.33.44+port=23456'
# first line is default DSA
# the next two lines must be on one line in the file
/CN=DirX-DSA-host3
TS=DSA3,NA='TCP/IP_IDM!internet=111.22.33.44+port=23333'
```

#### 7.2.2.2.2. Binding to the DSA with dirxcp

To bind to the bootstrapped DSA with **dirxcp**, the DSA3 administrator uses the command:

```
bind -authentication SIMPLE \
     -user /O=My-Company/OU=Manufacturing/CN=admin \
     -password dirx
```

#### 7.2.2.2.3. Creating the Access Control-Specific Subentry

Recall from the chapter “Setting up the DirX Directory Service” that the access control-specific subentry defines the access control policy for a subset of the administrative area with which it is associated. An access control specific subentry of the first administrative point must define the access control policy applicable in the DSA. By default, no user is granted any access right to any entry of the DIT.

The DSA3 administrator uses the following **dirxcp** command to creates the access control-specific subentry for My-Company’s Manufacturing organization below the first administrative point:

```
create \
         /O=My-Company/OU=Manufacturing/CN=AccessControl-Subentry \
         -attr \
            {OCL=SUBE;ACS} \
            {SS={DEF=TRUE}} \
            {PACI={ID=admin: enable most of operations \
                             but no Rename,
                   PR=254,
                   AL={BL={L=SIMPLE}},
         UF={UC={N={DN={/O=My-Company/OU=Manufacturing/CN=admin}};\
                   {DN={/O=My-Company/CN=admin}}},
                       UP={PI={E=TRUE},
                       GAD=grantDiscloseOnError+\
                           grantRead+grantBrowse+\
                           grantReturnDN+\
                           grantAdd+grantRemove+grantModify};
                       {PI={AT=AR;ACS;SACI;CRN;
                            CRT;MN;MT;SOC;GSR;CE;EACI,
                            AAV=AR;ACS;SACI;CRN;
                           CRT;MN;MT;SOC;GSR;CE;EACI,
                           AUATV=TRUE},
                      GAD=grantDiscloseOnError+\
                          grantRead+grantCompare+\
```

```
                        grantFilterMatch+\
                        grantAdd+grantRemove}} };
                   {ID=Public Access: enable Read and Search - \
                                      disable read password,
                    PR=0,
                    AL={BL={L=NONE}},
                    UF={UC={AU=TRUE},
                        UP={PI={E=TRUE},
                            GAD=grantDiscloseOnError+\
                                grantRead+\
                                grantBrowse+grantReturnDN};
                           {PI={AUATV=TRUE},
                            GAD=grantDiscloseOnError+\
                                grantRead+grantCompare+\
                                grantFilterMatch };
                           {PR=255,
                            PI={AT=UP},
                            GAD=denyRead} } } }
```

#### 7.2.2.3. Populating the DIT

To perform this task, the administrator of DSA3 follows the same procedures as described in the section “Populating the DIT” in the chapter “Setting up the DirX Directory Service”. The only difference for DSA3 is the distinguished name of the administrator in the bind operation that occurs prior to creating the entries in the DIT. For example:

```
create {/O=My-Company/OU=Manufacturing/CN=Klein Mike} \
        -attr \
            {OCL=TOP;PER;ORP;MUS} \
            {SN=Klein} \
            {DSC=Plant Manager} \
            {TN=+49 68 567 890;+49 68 567 891} \
            {UP=mdk123}
```

### 7.2.3. Establishing DSA Communications

Now My-Company has another stand-alone DSA running that holds the entries of the manufacturing organizational unit. My-Company employees can bind to the DSA with **dirxcp** and read these entries.

However, at this point in the process, DSA1 and DSA3 are two stand-alone DSAs that cannot communicate with each other and which have completely separate naming contexts. To

create the distributed DIT, the administrators of the two stand-alone DSAs need to establish transition points between each DSA's naming context. These transition points are called **knowledge references**, and contain information about the DSAs that hold the pieces of the distributed DIT.

In the case of My-Company, the naming context **/O=My-Company** on DSA1 has a lower border that includes the employee entries for the Sales and Development organizational units. The administrator on DSA1 needs to create a knowledge reference that extends this border to include the naming context **/O=My-Company/OU=Manufacturing** on DSA3. This knowledge reference is called a **subordinate reference** and represents the transition from the superior DSA's naming context (in this case, DSA1) to the subordinate DSA's naming context (in this case, DSA3). The subordinate reference contains enough information for the superior DSA to forward to the subordinate DSA any inquiries that relate to its naming context. For example, if a DUA binds to DSA1 and initiates a search operation below the Manufacturing organizational unit, DSA1 will forward, or **chain**, the request to DSA3 using the subordinate reference and the DSP protocol.

The administrator of the subordinate DSA must also create a knowledge reference to the superior DSA. This knowledge reference is called a **superior reference**, and enables the subordinate DSA to communicate with the superior DSA if it does not have enough information to answer a query.

Furthermore, as discussed in the chapter "Creating a Shadow DSA", DSAs must be protected from unknown or untrusted DSAs trying to connect to them for unknown purposes. Since the superior and subordinate DSAs will need to contact each other, each needs a mechanism to determine whether it is the appropriate DSA who is contacting it or whether it should reject the connection. As in shadowing, the mechanism used to screen connecting DSAs is the DSA policy.

Consequently, establishing DSA communications consists of these steps:

- Creating a DSA policy for each DSA that permits communication with the other
- Creating the superior and subordinate references on each DSA

These steps are performed in two stages:

1. Establish communication from the superior to subordinate DSA
2. Establish communication from the subordinate to the superior DSA

The next sections describe how to perform these tasks in the context of the sample distributed configuration.

#### 7.2.3.1. Establishing Communications from Superior to Subordinate DSA

This procedure consists of the following tasks:

1. Creating the DSA policy on the subordinate DSA
2. Changing the DSA password on the superior DSA
3. Creating the subordinate reference on the superior DSA

4. Testing the chaining from superior to subordinate DSA

#### 7.2.3.1.1. Creating the DSA Policy on the Subordinate DSA

The first step is for the administrator on DSA3 to create a DSA policy that contains the name and password of the DSA that is permitted to connect to DSA3, which in this case, is DSA1. This policy will enable DSA1 to chain to DSA3 if necessary.

Since chained operations can result in the modification of the subordinate DSA's local DIT or in the display of sensitive information, the DSA policy established here must also define the level of trust to be given to the chaining DSA. The policy that My-Company establishes on DSA3 defines DSA1 as a trusted DSA for read operations, but not for modify operations.

To create a DSA policy, use the **dirxadm** command:

[**dse**] **modify** / **-addattribute** *attribute_list*

The modify operation is performed on the root DSE because the DSA-policy structured attribute (DSAP) is an attribute of the root DSE.

For the DSA policy established on DSA3, the attribute list must contain the DSA-policy (DSAP) attribute with the following components and values:

- The DSA-name component (DSA) with the value **/CN=DirX-DSA-host1**, which is DSA1's name.
- The password (PWD) subcomponent of the authentication-policy (AP) component with the value **M8921DSA1**, which is DSA1's password.
- The authentication-method-trusted (AMT) subcomponent with the value **AM=SIMPLE-WITH-PWD;SIMPLE-PROTECTED**. This value means that if DSA1 uses a different authentication method when it contacts DSA3, for instance **NONE** or **SIMPLE**, DSA3 will not trust it. The connection, however, will be accepted. Not trusting DSA1 will result in operations being performed with the access rights of an anonymous user.
- The trusted-degree (TD) subcomponent with the value **READ**. This value means that DSA1 will only be trusted for chained read operations and not for chained modify operations.

The administrator on DSA3 creates this DSA policy with the following **dirxadm** command:

```
modify / -addattr {DSAP={DSA={/CN=DirX-DSA-host1},
          AP={AMT={AM=SIMPLE-WITH-PWD;SIMPLE-PROTECTED},
                              TD=READ,
                              PWD=M8921DSA1
                              }
                        }
                }
```

This command permits the DSA with the name **/CN=DirX-DSA-host1** and the user

password **M8921DSA1** to bind to DSA3. It also defines the authentication method for incoming trusted DSP binds to DSA3 so that only incoming DSP binds from DSA1 with simple credentials with password (protected or not) will be trusted, and defines the trust degree so that DSA1 is trusted for read operations but not for modify operations.

This example uses the attribute component AMT (authentication-method-for-trust) and does not use the AMB (authentication-method-for-bind) component. Not using AMB means that DSA3 will accept any authentication level for incoming DSP associations (see the description of the DSA policy syntax in the *DirX Directory Syntaxes and Attributes*).

Recall that the access control subentry on DSA3 gives the DSA1 administrator the same access rights on DSA3 as its local administrator. However, the level of trust established in DSA3's DSA policy affects these access rights. For example, suppose the DSA1 administrator wants to view the operational attributes of an entry or a subentry held by DSA3. To do this, the administrator binds to his DSA, which is DSA1, and issues a **dirxcp** request, for example:

```
show /O=My-Company/OU=Manufacturing -alloperational -p
```

The DSA1 administrator will receive the information because DSA1 will chain to the DSA3 and the request is a read operation, which the DSA policy allows.

However, should the DSA1 administrator try to modify the part of the DIT on DSA3, he will receive the message "insufficient access rights" because DSA1 is not trusted for modify operations, even though the access control subentry in DSA3 gives him these rights.

#### 7.2.3.1.2. Changing the Default DSA Password on the Superior DSA

Every DirX Directory DSA has after its installation the default password **DSA**; this value is hardcoded and not visible. Since My-Company has decided to use different passwords for its DSAs and the DSA policy for DSA3 has been set up to expect a password from DSA1 of **M8921DSA1**, the administrator on DSA1 needs to modify the DSA policy to contain this password.

To change a DSA password, use the **dirxadm** command:

[**dse**] **modify** / **-addattribute** *attribute_list*

For My-Company, the DSA1 administrator must specify the following:

- The DSA-policy (DSAP) attribute in *attribute_list* with the following values:
  - The **/** character in the DSA-name (DSA) component. Recall from the chapter "Setting up the DirX Directory Service" that the DSAP attribute can contain several types of policies (it is a multivalued attribute). If the name of a DSA is explicitly specified, the policy applies only to the named DSA. If "/" is specified as the DSA name, this policy applies to all other DSAs; that is, those DSAs for which no specific policy exists.
  - The password (PWD) component contains the value **M8921DSA1**, which is the password for DSA1

The following **dirxadm** command establishes DSA1's password:

```
modify / -addattr {DSAP={DSA={/},
                         AP={PWD=M8921DSA1}
                           }
                   }
```

#### 7.2.3.1.3. Creating a Subordinate Reference on the Superior DSA

To create a subordinate reference, use the **dirxadm** command:

[**dse**] **create** *distinguished_name* **-attribute** *attribute_list*

Supply the distinguished name of the subordinate context prefix in the *distinguished_name* argument. The attribute list must contain the following attributes:

- The DSET attribute with the value SUBR
- The specific-knowledge (SK) attribute, which specifies the access point of the subordinate DSA and its category

For My-Company, the administrator on DSA1 supplies the following values:

- The subordinate context prefix **/O=My-Company/OU=Manufacturing** in the distinguished_name argument
- The DSA name for DSA3 in the AE subcomponent of the master-or-shadow-access-point (MOS) component of the SK attribute
- The keyword **MASTER** in the category component of the SK attribute to indicate that DSA3 is not a shadow DSA but is the master for the entries in the Manufacturing portion of the DIT

The following **dirxadm** command creates the subordinate reference on DSA1:

```
create /O=My-Company/OU=Manufacturing  -attr \
                            DSET=SUBR \
  {SK={MOS={MSAP={AE={/CN=DirX-DSA-host3},
       PSAP={TS=DSA3,
          NA='TCP/IP_IDM!internet=111.22.33.44+port=23333'
           }
        },
       CAT=MASTER
       }
     }
   }
```

#### 7.2.3.1.4. Testing Communications from the Superior to the Subordinate DSA

The procedures given in the previous sections establish communication from the superior to the subordinate DSA in the sample distributed configuration. A possible next step is to verify that the superior DSA can access the DIT held by the subordinate DSA by binding to the superior DSA and issuing **dirxcp** requests that cause chaining to the subordinate DSA.

For My-Company, the previous steps have established the conditions for communication from DSA1 to DSA3. The administrator on DSA1 can now test this with the following procedure:

1. Bind to DSA1 as the default administrator or as anonymous user. For example:

```
bind -auth SIMPLE -user /O=My-Company/CN=admin -password dirx
```

2. Issue a **dirxcp** read operation on DSA3's DIT. For example:

```
list /O=My-Company/OU=Manufacturing -pretty

search /O=My-Company/OU=Manufacturing \
    -subtree -alluserattributes -p
```

The DSA1 administrator has not defined a policy for the authentication method for outgoing DSP associations (AMB). Consequently, the default will be used, which is simple protected. For simple protected, the default validity time for the authentication credentials is 30 seconds. Since DSA1 and DSA3 are on different systems that each has its own clock, the administrators must ensure that both clocks are well synchronized. If the clocks' time difference is more than 30 seconds, the DSP association will be rejected because the credentials have expired.

### 7.2.3.2. Establishing Communications from the Subordinate to the Superior DSA

This procedure consists of the following tasks:

1. Creating the DSA policy on the superior DSA
2. Changing the DSA password on the subordinate DSA
3. Creating the superior reference on the subordinate DSA

#### 7.2.3.2.1. Creating the DSA Policy on the Superior DSA

The administrator on DSA1 also needs to create a DSA policy that contains the name and password of the DSA that is permitted to connect to DSA1, which in this case, is DSA3. This policy will enable DSA3 to use DSP communications to DSA1 if necessary.

To create a DSA policy, use the **dirxadm** command:

[**dse**] **modify** / **-addattribute** *attribute_list*

The DSA1 administrator supplies the DSA-policy (DSAP) attribute in the *attribute_list*

argument with the following components and values:

- The DSA-name component (DSA) with the value **/CN=DirX-DSA-host3**, which is DSA3's name.
- The password (PWD) subcomponent of the authentication-policy (AP) component with the value **YY4321DSA3**, which is DSA3's password.

The following **dirxadm** command creates the DSA policy on DSA1:

```
modify / -addattr
{DSAP={DSA={/CN=DirX-DSA-host3},
          AP={PWD=YY4321DSA3}
       }
 }
```

The DSA policy for DSA1 does not define a level of trust. All users who bind to DSA3 and want information from DSA1 will receive the same rights as anonymous users, independent of whether or not they are authenticated. As a result, the local administrator of DSA3 is permitted to read the user attributes of the entries held on DSA1 but not the operational attributes, because this is the access control policy for anonymous users that has been established on DSA1.

#### 7.2.3.2.2. Changing the Default DSA Password on the Subordinate DSA

The administrator on DSA3 also needs to modify the DSA policy to contain the new DSA3 password. To change a DSA password, use the **dirxadm** command:

[**dse**] **modify / -addattribute** *attribute_list*

For My-Company, the DSA3 administrator must specify the following:

- The DSA-policy (DSAP) attribute in *attribute_list* with the following values:
  - The **/** character in the DSA-name (DSA) component; this value means that the policy will be used for all partner DSAs for which no special DSA policy exists that contains the exact DSA name
  - The password (PWD) component contains the value **YY4321DSA3**, which is the password for DSA3

The following **dirxadm** command establishes DSA3's password:

+

```
modify / -addattr {DSAP={DSA={/},
                         AP={PWD=YY4321DSA3}
                           }
```

```
                    }
```

#### 7.2.3.2.3. Creating a Superior Reference on the Subordinate DSA

To create a superior reference, use the **dirxadm** command:

[**dse**] **modify / -addattribute** *attribute_list*

and supply:

- The superior-knowledge (SPK) structured attribute in the *attribute_list* argument; this attribute specifies the access point and presentation address of the superior DSA. In the case of My-Company, this is the DSA name and presentation address of DSA1.

The modify operation is performed on the root DSE because the superior-knowledge (SPK) structured attribute is an attribute of the root DSE.

The following **dirxadm** command creates the superior reference on DSA3:

```
modify / -addattr \
         {SPK={AE={/CN=DirX-DSA-host1},
            PSAP={TS=DSA1,
              NA='TCP/IP_IDM!internet=123.45.56.89+port=19999'}
              }
         }
```

The DSA3 administrator can now bind to DSA3 and issue a **dirxcp** request for information about the entries in DSA1's part of the DIT, for example:

```
show /O=My-Company/OU=Sales -alluserattr -pretty
```

DSA3, using the superior reference, will chain to DSA1 to retrieve the information.The previous note about time synchronization between the two DSA machines applies here as well.

## 7.3. Extending My-Company Distributed DIT Configuration

This section of the chapter explains how to modify the sample distributed configuration established in the previous sections.It describes how to:

- Establish a DSA keep line policy
- Increase the expiration time of credentials for simple protected binds
- Use simple with password as authentication mechanism for DSP associations

The next sections describe the procedures to be followed to make these modifications to

the sample configuration using as an example My-Company.

### 7.3.1. Establishing a Keep-Line Policy

By default, every chained request from a DirX Directory DSA creates a new DSP association which is closed after the chained result is returned. If chained requests are sent frequently, administrators may want to change this default so that associations remain open across multiple chained requests.

The DSA keep-line policy permits administrators to specify a time limit during which a DSP association will remain open. Chained requests that occur within this time frame will use the same association. Administrators can set this policy for all DSAs (by specifying the DSA name **/**) or only for specific DSAs.

To create a keep line policy, use the **dirxadm** command:

[**dse**] **modify** / **-addattr** *attribute_list*

and supply the DSA-policy (DSAP) attribute in *attribute_list* with the following values:

- The DSA-name (DSA) component, which is either **/** or a DSA name
- The keep-line (KP) component, which specifies the time limit for the association

For example, suppose the DSA1 administrator wants to extend the DSA policy on DSA1 to include a keep-line policy of 60 seconds on all outgoing DSP associations from DSA1 to DSA3. The policy is to apply for all partner DSAs of DSA1. The administrator must modify the DSA policy for other DSAs in two steps:

1. Remove the old policy from the DSAP attribute (policies are one of the multiple values of the DSAP attribute)
2. Add the new policy and additionally specify any old options which should be kept, for example, the password

The following **dirxadm** commands create this keep-line policy:

```
modify / -removeattr DSAP={/}
modify / -addattr {DSAP={DSA={/},
                         AP={PWD=M8921DSA1},
                         KL=60
                           }
                  }
```

### 7.3.2. Increasing the Expiration Time of Credentials

By default, authentication credentials for simple protected DSP binds expire after 30 seconds. Since the two DSAs are on two different systems that each has its own clock, the administrators must ensure that both clocks are well synchronized. If the clocks' time

difference is more than 30 seconds, the DSP bind will be rejected because the credentials have expired.

There are two ways to increase the validity of credentials for simple protected binds to avoid clock synchronization problems between responder and initiator:

1. Increase the credentials validity on the responder side
2. Increase the credentials expiration time on the initiator side

The My-Company administrators choose to use the second approach.

The DSA authentication policy permits administrators to specify the expiration time of credentials for outgoing DSP (and DISP) binds. To create a DSA authentication policy, use the **dirxadm** command:

[**dse**] **modify / -addattr** *attribute_list*

and supply the DSA-policy (DSAP) attribute in *attribute_list* with the following values:

- The DSA-name (DSA) component, which is either **/** or a DSA name
- The authentication-policy (AP) component, which specifies (among others) the credentials' expiration time (CE)

For example, suppose the DSA1 administrator wants to extend the DSA policy on DSA1 to include a DSA authentication policy with a credentials expiration time of 3600 seconds (one hour) on all outgoing DSP (and DISP) associations from DSA1 to DSA3. The policy is to apply for all partner DSAs of DSA1. The administrator must modify the DSA policy for other DSAs in two steps:

1. Remove the old policy from the DSAP attribute (policies are one of the multiple values of the DSAP attribute)
2. Add the new policy and additionally specify any old options which should be kept, for example, the password and the keep-line policy.

The following **dirxadm** commands create this DSA authentication policy:

+

```
modify / -removeattr DSAP={/}
modify / -addattr {DSAP={DSA={/},
                         AP={PWD=M8921DSA1,
                             CE=3600},
                         KL=60
                            }
                  }
```

### 7.3.3. Using Simple-with-Password Authentication for Outgoing DSP Associations

By default, simple protected authentication is used for all outgoing DSP (and DISP) binds. However, the DSA authentication policy permits administrators to specify the authentication method to be used for outgoing DSP binds.

To create and modify a DSA authentication policy, use the **dirxadm** command:

[**dse**] **modify / -addattr** *attribute_list*

and supply the DSA-policy (DSAP) attribute in *attribute_list* with the following values:

- The DSA-name (DSA) component, which is either **/** or a DSA name
- The authentication-policy (AP) component, which specifies (among others) the authentication method for binds (AMB)

Recall that the DSA1 administrator did not specify an authentication method for binds when he set up DSA1; consequently, the default of simple protected is used. Now suppose he wants to extend the DSA policy on DSA1 to include a DSA authentication policy with simple-with-password authentication for all outgoing DSP (and DISP) associations from DSA1 to DSA3. The policy is to apply for all partner DSAs of DSA1.

The administrator must modify the DSA policy for other DSAs in two steps:

1. Remove the old policy from the DSAP attribute (policies are one of the multiple values of the DSAP attribute)
2. Add the new policy and additionally specify any old options which should be kept, for example, the password and the keep-line policy. Note that there is no need to keep a value for the expiration time of credentials, as this value is only relevant for simple protected authentication.

The following **dirxadm** commands create this DSA authentication policy:

```
modify / -removeattr DSAP={/}
modify / -addattr {DSAP={DSA={/},
                         AP={AMB={AM=SIMPLE-WITH-PWD},
                             PWD=M8921DSA1},
                         KL=60
                            }
                  }
```

# 8. Multireplication

This chapter describes how to set up a directory service based only on shadowing in which pieces of a DIT held by two different DSAs are replicated to each DSA.The chapter describes a sample shadow configuration with the following characteristics:

- The first DSA is the sample stand-alone DSA established in the chapter “Setting up the DirX Directory Service”.It holds the master My-Company entries that are replicated to the DSA established in the chapter “Distributing the DIT across Multiple DSAs“, and also holds shadow entries from this DSA.
- The second DSA is another stand-alone DSA that masters entries from another company.These entries are replicated to the first DSA.The second DSA also holds as shadow all entries from the first DSA.

This chapter continues to use the fictitious company “My-Company” to illustrate the planning decisions and administrative tasks necessary to set up this sample shadow configuration.

Tcl scripts for building this sample shadow configuration and adding the extensions to it can be found in the directory *install*]path/_**scripts/multireplication**.These scripts contain all the commands and procedures discussed in this chapter.

## 8.1. Planning the Shadow Configuration

My-Company has acquired a sales partner in the United States: the STU company in Boston, Massachusetts.My-Company has already established a single stand-alone DSA at the company’s Munich headquarters that contains the employee directory of its Sales and Development departments.The STU company has also established a stand-alone DSA in Boston that contains its company employee directory.

In order to optimize the contacts and speed the work coordination between the two companies, My-Company and STU decide that My-Company’s Sales and Development employee directories and STU’s entire employee directory should be available locally at each site.They also determine that My-Company’s Manufacturing employee directory, located on a DSA in Durach, need not be as highly available to the STU company as its Sales and Development directories.Consequently, the My-Company and STU companies decide to replicate My-Company’s Sales and Development employee directory on the DSA in Boston, and to replicate the STU company’s employee directory on the DSA in Munich.The following figure illustrates this solution.

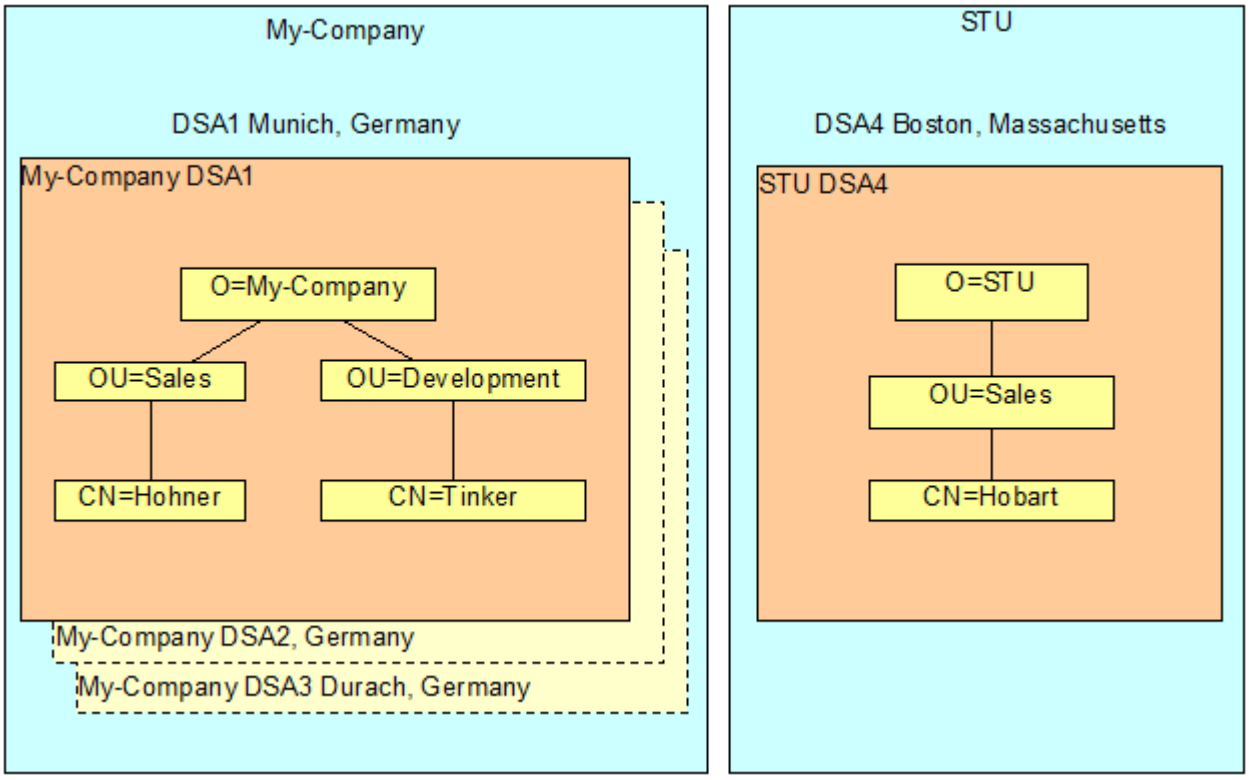



*Figure 30. My-Company-STU Shadow Configuration*

## 8.2. Building the Shadow Configuration

Recall from the chapter “Creating a Shadow DSA” that a DSA that holds the information to be replicated is the supplier DSA, while the DSA that receives the information is the consumer DSA.In this case, each DSA plays both supplier and consumer roles:

- DSA1 is a supplier of DSA4, since it holds the master My-Company employee entries that will be replicated to DSA4, and is also a consumer of DSA4, since it holds shadows of the STU employee entries that DSA4 masters.
- DSA4 is a supplier of DSA1, since it holds the master STU employee entries that will be replicated to DSA1, and a consumer of DSA1, since it holds shadows of the My-Company Sales and Development entries that DSA1 masters.

Recall from section “Shadowing in a Heterogeneous Network” in the chapter “Creating a Shadow DSA“, that DirX Directory supports two kinds of shadowing: central shadowing administration (CENTRALADMIN) and X.500 compliant shadowing (X500).Central shadowing administration is easier to handle than X.500 compliant shadowing because shadowing agreements are administrated once on a single DSA and distributed by shadowing itself.My-Company’s DSA1 is already the DSA for the central shadowing administration.STU also uses DirX Directory and DSA4 is not yet part of any other central shadowing.The administrators decide to connect DSA4 to the already established central shadowing.

The administrator for DSA1 must therefore create two shadowing agreements:

- A shadowing agreement in which DSA1 is the supplier and DSA4 is the consumer and in which the information to be replicated is the **Sales** and **Development** organizational-

units of the subtree **/O=My-Company**.The **Manufacturing** employee subtree **OU=Manufacturing**, which resides on DSA3 in Durach, will not be replicated to the STU DSA.

- A shadowing agreement in which DSA4 is the supplier and DSA1 is the consumer and in which the information to be replicated is the subtree **/O=STU**.

The administrator can perform these steps locally on his DSA by binding to the DSA with the **dirxadm bind** command and then issuing **dirxadm** commands.Alternatively, the administrators can use DirX Directory Manager from a Windows system to bind to DSA1 to perform these tasks.The next sections describe how to perform these tasks with **dirxadm**.

# 8.3. Negotiating the Shadowing Agreements

The administrators of DSA1 and DSA4 must first establish the details of the two shadowing agreements between their two DSAs.These details include:

- Determining the DSA roles and identifying the names, presentation addresses, and passwords of each DSA
- Determining the units of replication
- Determining the shadowing agreement Ids
- Determining the update strategy

## 8.3.1. Determining the DSA Roles

For My-Company in Munich, DSA1 is to be both a supplier DSA and a consumer DSA. Its distinguished name is **/CN=DirX-DSA-host1** and it has the following presentation address:

| | | |
|---|---|---|
| Transport selector: | | DSA1 |
| Network address: | Internet address: | 123.45.67.89 |
| | port number: | 19999 |

For STU in Boston, DSA4 is to be both a supplier DSA and a consumer DSA. Its distinguished name is **/CN=DirX-DSA-host4**, and it has the following presentation address:

| | | |
|---|---|---|
| Transport selector: | | DSA4 |
| Network address: | Internet address: | 111.22.33.55 |
| | port number: | 24444 |

The password for DSA1 is **M8921DSA1** and the password for DSA4 is **B4321DSA4**.

## 8.3.2. Determining the Unit of Replication

For My-Company in Munich, the unit of replication is the entire naming context **/O=My-Company**. All entries within this naming context will be duplicated and all attributes will be duplicated. (The Organizational-Unit **Manufacturing** is a separate naming context that resides in another DSA).

For STU in Boston, the unit of replication is the entire naming context **/O=STU**, which will be replicated to the My-Company DSA in Munich (DSA1).

### 8.3.3. Determining the Agreement Ids

In this shadow configuration, DSA1 and DSA4 share two agreements: one in which the partner DSA is the consumer, and one in which the partner DSA is the supplier.The administrators of DSA1 and DSA4 must choose unique identifiers for each of these agreements.The administrators select:

- ID **14,0** to identify the shadowing agreements in which DSA1 is the supplier and DSA4 is the consumer
- ID **41,0** to identify the shadowing agreements in which DSA1 is the consumer and DSA4 is the supplier

After creating these shadowing agreements on DSA1 they are distributed to DSA4 by shadowing itself.

## 8.4. Using dirxadm to Prepare DSA4 for Shadowing

Now the two administrators have negotiated the details of the shadowing agreements.Even though that the shadowing agreements are all administered on My-Company's DSA1 the administrator of STU's DSA4 must prepare his DSA for shadowing operations.So his next step is to use **dirxadm** to:

1. Perform an authenticated administrator bind to DSA4, which is the local DSA
2. Create a DSA policy on DSA4 that allows DSA1 to initiate a DISP communication with DSA4.
3. Change the default password on DSA4.

### 8.4.1. Binding with Authentication to DSA4

In the process of setting up the STU stand-alone DSA, the STU administrator has:

- Established himself as the default administrator with the distinguished name **/O=STU/CN=admin**
- Selected to use simple authentication using the password **dirx**
- Added his distinguished name to the DirX Directory administrators attribute on DSA4 to allow him to bind with **dirxadm**

The STU administrator must now bind with authentication to DSA4 in order to use **dirxadm** to set up the DSA for shadowing.

To bind to a local DSA, use the **dirxadm** command:

**bind -user** *username* **-password** *password* **-authentication** *auth_]method*

To bind with simple authentication to the local DSA, which in this case is DSA4, the STU

administrator issues the **dirxadm** command:

```
bind -user /O=STU/CN=admin -password dirx -auth SIMPLE
```

### 8.4.2. Creating the DSA Policy on DSA4

Recall from the chapter “Creating a Shadow DSA” that DSAs must be protected from unknown or untrustworthy DSAs trying to connect to them for unknown purposes. When the administrator for DSA1 establishes the shadowing agreement, DSA1 will try to contact DSA4 to start the total update. Consequently, DSA4 needs a mechanism to identify whether the connecting DSA is DSA1 or whether it should reject the connection.

The mechanism used to screen connecting DSAs is the DSA policy, which contains the names and passwords of all DSAs permitted to connect to a given DSA. To enable DSA4 to recognize DSA1, the administrator of DSA4 creates a DSA policy that contains DSA1’s name and password.

To create a DSA policy, use the **dirxadm** command:

dirxadm> [**dse**] **modify / -addattribute** *attribute*]list_

The modify operation is performed on the root DSE because the DSA-policy structured attribute (DSAP) is an attribute of the root DSE.

The attribute list must contain the DSA-policy (DSAP) attribute with the following components:

- The DSA-name component (DSA), which in this example has the value **/CN=DirX-DSA-host1**, which is DSA1’s name.
- The password (PWD) subcomponent of the authentication-policy (AP) component which in this example has the value **M8921DSA1**, which is DSA1’s password.

The following **dirxadm** command creates **STU-DSA4**’s DSA policy:

```
modify / -addattr \
             {DSAP={DSA={/CN=DirX-DSA-host1},   \
                   AP={PWD=M8921DSA1}
               }
         }
```

### 8.4.3. Changing the Default Password on DSA4

Recall from the chapter "Creating a Shadow DSA" that every DirX Directory DSA has, after its installation, the default password **DSA**; this value is hard-coded and is not visible. Since the My-Company and STU administrators have decided to use different passwords for their DSAs and the DSA policy for DSA1 will be set up to expect a password from DSA4 of **B431DSA4**, the administrator on DSA4 needs to modify the DSA policy to contain this password.

To change a DSA password, use the **dirxadm** command:

[**dse**] **modify** / **-addattribute** *attribute*]list_

The STU DSA4 administrator must specify the following:

- The DSA-policy (DSAP) attribute in *attribute*]list_ with the following values:
  - The **/** character in the DSA-name (DSA) component. The DSAP attribute can contain several types of policies (it is a multivalued attribute). If the name of a DSA is explicitly specified, the policy applies only to the named DSA. If "/" is specified as the DSA name, this policy applies to all other DSAs; that is, those DSAs for which no specific policy exists.
  - The password (PWD) component contains the value **B4321DSA4**, which is the password for DSA4

The following **dirxadm** command establishes DSA4's password:

```
modify / -addattr {DSAP{DSA={/},
                       AP={PWD=B4321DSA4}
                  }
```

# 8.5. Using dirxadm to Prepare DSA1 for Shadowing

The administrator of My-Company's DSA1 must perform the same procedures on DSA1 as the STU DSA4 administrator has done on DSA4. Since the My-Company DSA1 administrator has already changed DSA1's default password to **M8921DSA1** when he created the shadowing agreement with DSA2 described in the chapter "Creating a Shadow DSA", he need only:

1. Perform an authenticated administrator bind to DSA1, which is the local DSA
2. Create a DSA policy on DSA1 that allows DSA4 to initiate a DISP connection with DSA1.
3. Create two shadowing agreements on DSA1, one for the consumer role and one for the supplier role.

## 8.5.1. Binding with Authentication to DSA1

In the process of setting up the My-Company stand-alone DSA **DirX-DSA-host1** described in the section "Setting up the DSA" in the chapter "Setting up the DirX Directory Service", the My-Company administrator has:

- Established himself as the default administrator with the distinguished name **/O=My-Company/CN=admin**
- Selected to use simple authentication using the password **dirx**
- Added his distinguished name to the DirX Directory administrators attribute on DSA1 to allow him to bind with **dirxadm**

The My-Company administrator must now bind with authentication to DSA1 in order to use **dirxadm** to set up the DSA for shadowing.

To bind with simple authentication to the local DSA, which in this case is DSA1, the My-Company administrator issues the **dirxadm** command:

```
bind -user /O=My-Company/CN=admin -password dirx -auth SIMPLE
```

## 8.5.2. Creating the DSA Policy on DSA1

To create the DSA policy on DSA1, the My-Company DSA1 administrator uses the same **dirxadm** command as the STU DSA4 administrator used on DSA4:

dirxadm> [**dse**] **modify** / **-addattribute** *attribute*]list_

For DSA1, the attribute list must contain the DSA-policy (DSAP) attribute with the following components and values:

- The DSA-name component (DSA) with the value **/CN=DirX-DSA-host4**, which is DSA4's name.
- The password (PWD) subcomponent of the authentication-policy (AP) component with DSA4's password **B4321DSA4**.

The following **dirxadm** command creates **My-Company-DSA1**'s DSA policy:

```
modify / -addattr \
             {DSAP={DSA={/CN=DirX-DSA-host4},
                    AP={PWD=B4321DSA1}
               }
         }
```

## 8.5.3. Creating the Shadowing Agreements on DSA1

To create the two shadowing agreements on DSA1, the My-Company DSA1 administrator uses the **dirxadm sob create** command.

### 8.5.3.1. Creating the Agreement for O=My-Company Shadowed From DSA1 to DSA4

To create the shadowing agreement on DSA1 that describes DSA1 as a supplier and DSA4 as a consumer, the administrator supplies the following values:

- The name of DSA4 to the **-consumer** option
- The presentation address of DSA4 to the **-consumerpsap** option
- The keyword **CENTRALADMIN** to the -**consumerkind** option, which specifies a DirX-Directory-compliant/asynchronous shadowing
- The agreement identifier **14,0**

- The keyword **COOPERATIVE** to the **-status** option, which means that DSA1 will be ready to send a total update to DSA4 immediately
- The shadowing-subject (SS) operational attribute to the **-agreement** option, which defines:
  - The name of the context prefix (CP) to be replicated in the AREA component. In this agreement, it is DSA1's context prefix **/O=My-Company**.
  - The entries belonging to this naming context to be replicated in the replication-area (RA) subcomponent. In this agreement, the value is **DEF=TRUE**, which means that the whole naming context must be replicated.
  - The attributes to be replicated in the ATT subcomponent. In this agreement, the value is **DEF=TRUE** to indicate that all attributes are to be replicated.
  - The update mode in the update-mode (UM) component. In this agreement, the value is **SI** (supplier-initiated).
  - The update strategy in periodic-strategy (PS) of the update-mode (UM) component. In this agreement, the strategy specifies a one-hour window in which DSA4 will accept updates (specified as 3600 seconds in the window-size (WS) subcomponent and schedules subsequent incremental updates once every 24 hours (specified as 82800 seconds in the update-interval (UI) subcomponent)

The administrator can omit the **-supplier** option (the name of DSA1) and the **-supplierpsap** option (presentation address of DSA1) because DSA1 is the supplier DSA and the administrator is bound to DSA1.

The following **dirxadm** command creates this shadowing agreement on DSA1:

```
sob create
-consumer {/CN=DirX-DSA-host4 }     \
-consumerpsap "TS=DSA4,NA='TCP/IP!internet=111.22.33.55+port=24444'"
\
-consumerkind CENTRALADMIN \
-agreementid 14,0       \
-status COOPERATIVE \
-agreement {SS={AREA={CP={/O=My-Company},
                      RA={DEF=TRUE}
                     },
                 ATT={DEF=TRUE}
                }
           },          \
           UM={SI={S={PS={WS=3600,
                          UI=82800
                         }
                     }
                }
```

```
                }
```

#### 8.5.3.2. Creating the Agreement for O=STU Shadowed From DSA4 to DSA1

To create the agreement that describes DSA1 as a consumer and DSA4 as a supplier, the administrator supplies the following values. (Keep in mind that this agreement is also created on DSA1.):

- The name of DSA4 to the **-supplier** option
- The name of DSA1 to the **-consumer** option
- The agreement identifier **41,0**
- The keyword **NONCOOPERATIVE** to the **-status** option, which postpones the first total update from DSA4 to DSA1 until the DSA1 administrator explicitly activates it by establishing the shadowing agreement.
- The shadowing-subject (SS) operational attribute to the **-agreement** option, which defines:
  - The name of the context prefix (CP) to be replicated in the AREA component. For this agreement, it is DSA4's context prefix, which is **/O=STU**.
  - The entries belonging to this naming context to be replicated in the replication-area (RA) subcomponent. For this agreement, the value is **DEF=TRUE**, which means that the whole naming context must be replicated.
  - The attributes to be replicated in the ATT subcomponent. For this agreement, the value is **DEF=TRUE** to indicate that all attributes are to be replicated.
  - The update mode and periodic update strategy in the update-mode (UM) component. For this agreement, the update mode is supplier-initiated, with incremental updates occurring every 24 hours and a one-hour window during which DSA1 will accept the updates.

The administrator can omit the **-supplierpsap** option (presentation address of DSA4) because this attribute was already administered when creating the agreement where DSA4 is consumer. He also can omit the **-consumerpsap** option because this attribute was already administered.

The following **dirxadm** command creates this shadowing agreement on DSA1:

```
sob create
-supplier {/CN=DirX-DSA-host4 }     \
-consumer {/CN=DirX-DSA-host1 }      \
-agreementid 41,0       \
-status NONCOOPERATIVE \
-agreement {SS={AREA={CP={/O=STU},
                      RA={DEF=TRUE}
                     },
```

```
                ATT={DEF=TRUE}
               }
            },          \
            UM={SI={S={PS={WS=3600,
                        UI=82800
                           }
                       }
                   }
               }
```

## 8.6. Using dirxadm to Activate the Shadowing Agreements

When the administrator created the shadowing agreements that described DSA1's consumer role, the administrator postponed shadowing activation by specifying the keyword **NONCOOPERATIVE** in the **-status** option to the **dirxadm sob create** command.Both administrators must now explicitly activate the shadowing agreement with the **dirxadm** command:

**sob establish –agreementid** *agreementid* **-consumer** *dsa*]name_

My-Company's DSA1 administrator uses the following **dirxadm** command to start the total update from DSA4 to DSA1:

```
sob establish -agreementid 41,0      \
-consumer {/CN=DirX-DSA-host4 }
```

## 8.7. Setting DUA Service Controls for Binds to Consumer DSAs

As described in the chapter "Creating a Shadow DSA", DUAs that bind to a consumer DSA to get local shadowed information from it should set two service controls:

- dontUseCopy to **FALSE**; if the value is TRUE, the consumer DSA understands that the DUA wants master Information and chains to the supplier DSA.
- copyShallDo to **TRUE**, which means that the consumer DSA will perform a query even if some information is missing in the shadow copy.If the consumer DSA is not able to completely satisfy the query, it will set IncompleteEntry in the returned entry information.

When using **dirxcp** as a DUA, these service controls can be set with the following **dirxcp** command:

**args modify -dontusecopy FALSE -copyshalldo TRUE**

# 9. Creating a Synchronous Shadow DSA

Recall from the section “Replication Services” in the chapter “Introducing DirX Directory” in the *DirX Directory Introduction* that DirX Directory supports two types of replication protocol for shadowing configurations:

- Asynchronous shadowing, which returns a DAP or LDAP client’s update operation immediately after the master DSA commits the operation. Asynchronous shadowing protocol supports highly flexible replication area definitions and allows a high rate of update operations between supplier and consumer DSAs, even over long distances, but can lead to loss of recent update operations at the consumer DSAs if the master DSA fails.
- Synchronous shadowing, which returns a DAP or LDAP client’s update operation only after the master DSA and all synchronous consumer DSAs have committed the operation. Synchronous shadowing protocol prevents data loss in the event of master DSA failure, but supports a lower rate of update operations between supplier and consumer DSAs and limits replication area flexibility.

The chapter “Creating a Shadow DSA” describes how to set up a floating-master shadow configuration that uses the asynchronous type of DirX Directory replication protocol. This chapter describes how to set up a floating-master shadow configuration that uses both asynchronous and synchronous shadowing protocols.

The sample shadow configuration described in this chapter builds on the sample scenarios described in the chapters “Creating a Shadow DSA”, “Distributing the DIT Across Multiple DSAs” and “Multireplication”. We suggest you read these chapters to familiarize yourself with shadowing, distribution and replication concepts and procedures before reading this chapter.

This chapter continues to use the fictitious company “My-Company” to illustrate the planning decisions and administrative tasks necessary to set up this sample shadow configuration.

## 9.1. Understanding Asynchronous and Synchronous Shadowing

Asynchronous and synchronous shadowing protocols each have benefits and drawbacks that must be evaluated before deciding whether and how to use them in a floating-master shadow configuration.The next sections describe these issues.

### 9.1.1. Understanding Asynchronous Shadowing

As mentioned in the section “Replication Services” in the chapter “Introducing DirX Directory” in the *DirX Directory Introduction*, asynchronous shadowing allows for complete flexibility with regard to defining the replication area, including attribute and object class selection, the type of updates performed, the update mode, and other aspects of configuration.The protocol supports high update operation rates, even at long distances between supplier and consumer DSA; however, if a supplier DSA outage occurs, recent

update operations may be lost at consumer DSAs.

Recall from the chapter “Creating a Shadow DSA” that at the time of total update, the DSA replicates entries and their attributes starting with the context prefix or a subtree below it as specified in the shadowing agreement information.After the total update occurs, every update in the tree that matches the specific shadowing agreement information initiated via DAP, DSP or DISP is also replicated.Recall, too, that changes made via RPC are not replicated; this means that changes made with **dirxadm**, such as deletion, creation and modification of entries in the tree and changes to audit and database configuration are not replicated to consumer DSAs.

The following figure illustrates asynchronous shadowing protocol flow.In the example, the LDAP client is connected to DSA1.

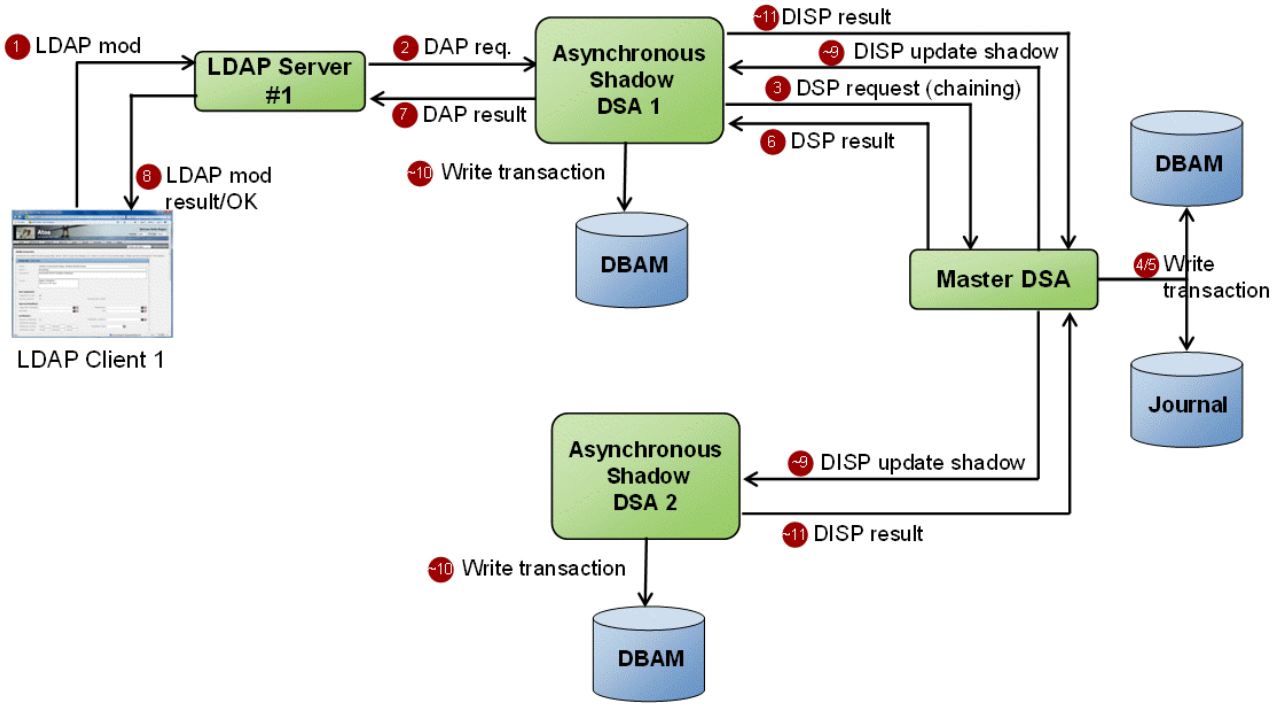



*Figure 31. Asynchronous Shadowing Protocol Flow*

As shown in the figure:

- In step 1, the LDAP client transmits the LDAP modify operation request to the LDAP server.
- In step 2, the LDAP server transmits the DAP modify operation request to shadow DSA1.
- In step 3, shadow DSA1 transmits a DSP request to the master DSA.
- In steps 4 and 5, the master DSA performs the modify operation, writes the changes into the translog device and then into the journal.
- In step 6, the master DSA transmits the DSP result to shadow DSA1.
- In step 7, shadow DSA1 transmits the DAP result to the LDAP server.
- In step 8, the LDAP server transmits the LDAP result to the LDAP client.

- Steps ~9 through ~11 are processed simultaneously:
  - In step ~9, the master DSA transmits the DISP incremental shadow update request to the shadow DSAs. The update shadow can contain multiple incremental update operations.
  - In step ~10, the shadow consumer DSAs perform the update operation(s) locally and then write the changes into their translog device.
  - In step ~11, the shadow consumer DSAs transmits the DISP results to the master DSA. This transmission can be delayed since it is possible to acknowledge multiple update operations with one PDU.

As you can see from this example, the LDAP client receives the result after the master DSA has committed the transaction but before the asynchronous shadow consumer DSAs have performed this operation locally. This principle is good for high update operation rates, but recent update operations can be lost at the asynchronous consumer DSAs if a master DSA outage occurs.

### 9.1.2. Understanding Synchronous Shadowing

Synchronous shadowing is designed to replicate the complete DIT of a DirX Directory service to full shadow DSAs in a fast and efficient way. The underlying protocol is synchronous: a DAP or LDAP client receives the result of an update operation only after the master DSA and all of the synchronous consumer DSAs have committed it. Synchronous shadowing operates at a lower update operations rate than asynchronous shadowing in favor of synchronized data. The main advantages of synchronous replication are:

- There is zero data loss in the event of master DSA outage.
- Data safety is always ensured. DAP/DSP/LDAP update operation is returned after master DSA and synchronous consumer DSAs have committed transaction. A read operation (that is search, read, list or compare request) after update operation delivers correct result if DAP/LDAP client is connected either to the master DSA or to a synchronous consumer DSA.
- Data synchronicity management delivers the correct search result as long as no network error occurs. If, for any reason, a synchronous consumer DSA's data is not synchronized with the master DSA's data, the read operation is automatically chained to the master DSA.

Synchronous shadowing agreement configuration offers fewer options than asynchronous shadowing:

- The complete DIT starting at the root DSE (/) and including all object classes and all attributes must always be replicated. The **-agreement** option in the **dirxadm sob create** command is not used. (See the **dirxadm sob create** operation in the *DirX Directory Administration Reference* for details.)
- Incremental changes must always be replicated on change. Synchronous shadowing does not permit scheduled agreements.
- The total update can be propagated with DISP or by media. However, if a total update is required - for example, after an error recovery - it is always performed via DISP

automatically without administrator intervention and independently from the configured settings.

- The automatic total update parameter specified in SOB-Policies should always be set to **TRUE**. We recommend this setting because it guarantees unattended data adjustment from supplier DSA to consumer DSA after an emergency switch.
- The **REPLS** subcomponent of the consumer policy must always be set to **TRUE**.

Although synchronous shadow agreement configuration is limited, DirX-Directory-compliant asynchronous shadowing agreements like the samples discussed in the chapter "Creating the Shadow DSA" (the *disp_type* value in the **–consumerkind** option is **CENTRALADMIN**) and X.500-compliant shadowing agreements (the *disp_type* value in the **–consumerkind** option is **X500**) can co-exist in a shadowing configuration as long as different DSAs are affected and the **REPLS** subcomponent of the DirX-Directory-compliant/asynchronous shadowing agreement is always **FALSE**; that is, you can administer DirX-Directory-compliant/asynchronous shadowing agreements and X.500 shadowing agreements for consumer DSAs if they don't participate in synchronous shadowing.

At the time of total update, the DSA replicates the complete DIT, starting with the root DSE, to all consumer DSAs that participate in synchronous shadowing. After the total update occurs, every update in the tree initiated via DAP, DSP and LDAP is also replicated. As with asynchronous shadowing, changes made via RPC are not replicated; this means that changes made with **dirxadm**, such as deletion, creation and modification of entries in the tree and changes to audit and database configuration are not replicated to consumer DSAs.

#### 9.1.2.1. How Synchronous Shadowing Works

The following figure illustrates synchronous shadowing protocol flow. In the example, the LDAP client is connected to DSA1.

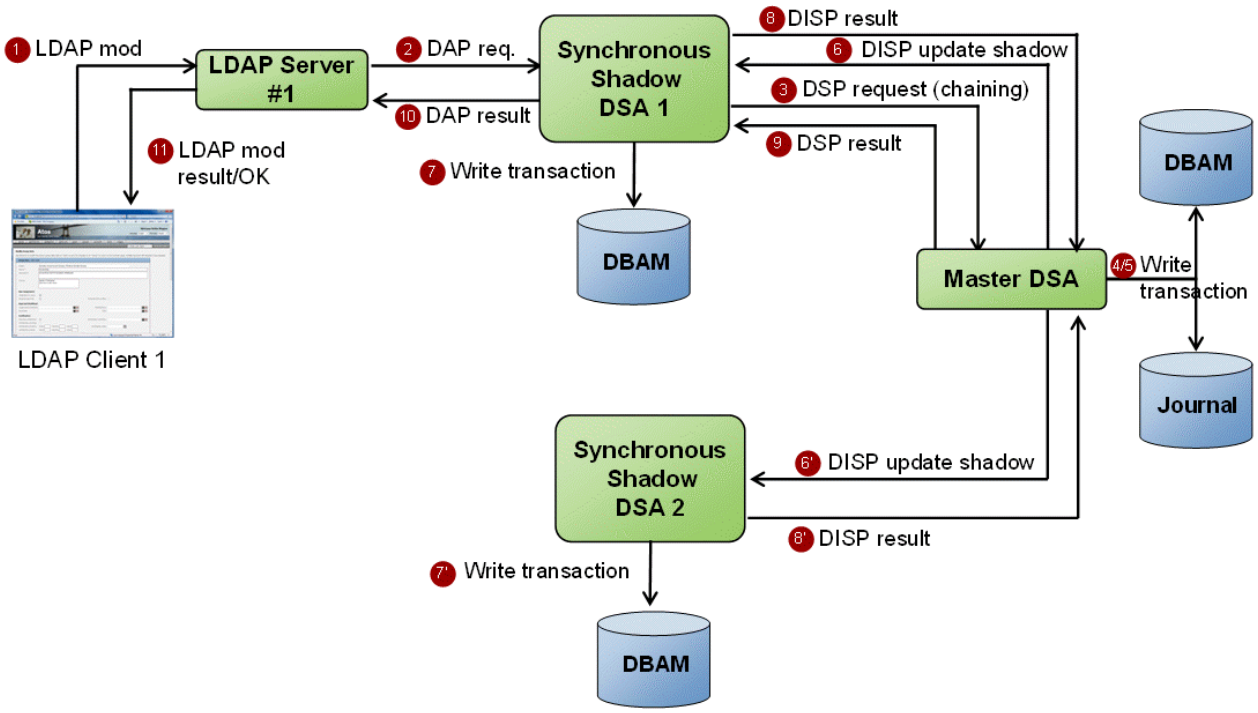



*Figure 32. Synchronous Shadowing Protocol Flow*

As shown in the figure:

- In step 1, the LDAP client transmits LDAP modify operation request to the LDAP server.
- In step 2, the LDAP server transmits DAP modify operation request to shadow DSA1.
- In step 3, shadow DSA1 transmits a DSP request to the master DSA.
- In steps 4 and 5, the master DSA performs a modify operation, writes changes into the translog device and then into the journal.
- In steps 6 and 6', the master DSA transmits the DISP update shadow to the shadow DSAs.
- In steps 7 and 7', the shadow DSAs perform the modify operation locally and then write changes into their transaction logs.
- In steps 8 and 8', the shadow DSAs transmit the DISP result PDU to the master DSA.
- In step 9, the master DSA transmits the DSP result to the shadow DSAs. The DSP result is successful independent of the results of the shadow DSAs because the update operation performed without error at the master DSA.
- In step 10, shadow DSA1 transmits the DAP result to the LDAP server.
- In step 11, the LDAP server transmits the result to the LDAP client.

As you can see from this example, the LDAP client receives the result after the master DSA and all synchronous shadow consumer DSAs have committed the transaction. This principle is good for zero data loss in the event a master DSA outage occurs, but it operates at lower operation rates compared to asynchronous shadowing.

#### 9.1.2.2. How Data Synchronicity is Managed

Another aspect of synchronous shadowing protocol is the status of data synchronicity at the consumer DSAs. To manage this situation, the master DSA assigns a modification sequence number (MSN) to each update operation—an add, delete modify or modifyDN request—made in the master DSA. The MSN is incremented with every update operation and is stored in the following locations:

- The recent MSN, which always keeps the highest MSN. The recent MSN is stored in the operational attribute DirX Directory Recent MSN (dirxRecentMSN) in the global subentry **/CN=DirXDBVersionSubentry**. This subentry and its operational attributes are also administered by the synchronous consumer DSAs.
- The entry MSN, whose value equals the value of the recent MSN at the time of the update operation. The entry MSN is stored in the created/modified entry in the operational attribute (DirX Directory Entry MSN (dirxEntryMSN)).
- The operation MSN, whose value equals the value of the entry MSN. The operation MSN is stored in the journal along with the update operation.
- The delta MSN, which is the difference between the recent MSN and the operation MSN calculated at the time at which the master DSA transmits the incremental update to the synchronous consumer DSA and is also part of the update PDU in addition to the update request. A value of 0 indicates a synchronous consumer DSA's data is synchronized with the master DSA's data.

A synchronous consumer DSA uses the following mechanisms to manage its data synchronicity status:

- When it receives a total update, it sets its data synchronicity status to **FALSE**.
- When it receives an incremental update, it sets its data synchronicity status depending on the value of the delta MSN: a value of 0 sets data synchronicity to **TRUE**, while any other value sets data synchronicity to **FALSE**.
- It polls the value of the recent MSN at the master DSA and then compares it with its own value. A difference between master and consumer values sets the data synchronicity status to **FALSE**.

When a synchronous consumer DSA starts to receive a total update, it sets its data synchronicity status to **FALSE**. Once the total update has completed, it changes its data synchronicity status to **TRUE** if no outstanding incremental updates exist. If outstanding incremental updates exist, the master DSA starts to transfer them. When the synchronous consumer DSA receives a delta MSN with the value 0, it changes its data synchronicity status to **TRUE**.

Administrators can determine a synchronous consumer DSA's current synchronicity status with the operational attribute DirX Directory In Sync (dirXInSync). This value is stored in the subentry **/CN=DirXDBVersionSubentry** and is always maintained by the DSA when it is part of a synchronous shadowing configuration. The administrator can use the **dirxadm sob show** command on the master DSA to display the current value of DirX Directory In Sync for a synchronous consumer DSA. The section "Monitoring Data Synchronicity Status" provides an example in the context of the My-Company synchronous shadowing scenario.

Administrators can monitor changes to a consumer DSA's synchronicity status in the following places:

- In the DSA's log file
- In the DSA's audit file
- Via SNMPv2 traps

The chapter "Monitoring DirX Directory" provides more information about these monitoring tools, and the section "Monitoring Data Synchronicity Status" in this chapter provides some examples.

#### 9.1.2.3. How Data Synchronicity Status Affects Read Operation Service Controls

In synchronous shadowing, a read operation from a synchronous consumer DSA is as good as from the master DSA if the synchronous consumer DSA's data synchronicity status is **TRUE**. Thus, the data synchronicity status impacts a synchronous consumer DSA's search engine by overriding the **COPYSHALLDO** service control. The following table shows this behavior when the **DONTUSECOPY** service control is **FALSE** and the **LOCALSCOPE** service control is **FALSE**:

| COPYSHALLDO Value | Synchronous Consumer DSA Synchronicity Status | Consumer DSA to Master DSA Bind Status Synchronization | Origin of Read Operation |
|---|---|---|---|
| TRUE | TRUE | Connected | Sync. consumer |
| TRUE | FALSE | Connected | Master via DSP **b)** |
| TRUE | TRUE **a)** | Not connected | Sync. consumer |
| TRUE | FALSE | Not connected | Sync. consumer |
| FALSE | TRUE | Connected | Sync. consumer **c)** |
| FALSE | FALSE | Connected | Master via DSP |
| FALSE | FALSE | Not connected | Error **d)** |
| FALSE | TRUE **a)** | Not connected | Sync. consumer **c)** |

As shown in the table:

a. If the network connection to the master DSA fails, the synchronous consumer DSA has compared its recent MSN value with that of other synchronous consumer DSAs. If its value is not lower than the value of any other synchronous consumer DSA, its data synchronicity status remains **TRUE**; otherwise, its data synchronicity status becomes **FALSE**.

b. The service control is overruled. An asynchronous consumer DSA would return the result of the read operation from the consumer DSA. A synchronous consumer DSA chains a read operation to the master DSA since its data synchronicity status is set to **FALSE** and could potentially return inaccurate results.

c. The service control is overruled. An asynchronous consumer DSA would chain the read

operation to the master DSA. A synchronous consumer DSA performs the read operation locally since its data synchronicity status is set to **TRUE** and returns accurate results as a master DSA would do.

d. The synchronous consumer DSA returns an LDAP UNAVAILABLE error.

As you can see from the information in the table, as long as there is no network failure, a DAP or LDAP client that is connected to a synchronous consumer DSA always sees an accurate search result.

#### 9.1.2.4. How Paging is Handled in a Synchronous Shadowing Environment

Based on the synchronicity status and the service control configuration, paged search operations can be executed on the consumer DSA or chained to the master via DSP. See the section “How Data Synchronicity Status Affects Read Operation Service Controls” for details. A resume information, which stores information about where to continue when the next page is requested is generated in the DSA that executes the first page request. As the resume information is stored in the memory of the DSA, no other DSA has any knowledge of it, so next page requests cannot be successfully executed on other DSAs. To ensure that subsequent next page requests are executed on the originator of the paged search operation, the DSA stores its role in the generated query reference. Based on this query reference, actual synchronicity status will be overridden and the request will be executed on the originator DSA.

As DSA roles are swapped when a switch occurs, generated query references would contain an incorrect role. Consequently, stored query references are invalidated on sob switch operations in the involved DSAs. See the description of “sob switch” in the *Administration Reference* for details.

#### 9.1.2.5. How Errors are Processed at the Consumer DSA

If the synchronous consumer DSA can’t perform an update operation due to some local error, it returns a shadow error to the master DSA and sets its data synchronicity status to **FALSE**. Depending on the service control settings, subsequent read operations are automatically chained to the master DSA to return an accurate result. The master DSA starts its error recovery procedure as defined in the shadowing initiator policies. If the error at the synchronous consumer is a temporary one (for example, a memory shortage) then it may not appear in subsequent re-tries and incremental updates can be performed again successfully. When the synchronous consumer DSA’s data becomes synchronized again, read operations are performed locally without administrator intervention. If the error is permanent, administrator intervention is required.

#### 9.1.2.6. How Errors and Timeouts are Processed at the Master DSA

The master DSA keeps track of the data synchronicity status of each synchronous consumer DSA. After a synchronous shadow agreement is created, the master DSA first propagates a total update to the synchronous consumer DSA and then subsequently propagates all outstanding incremental updates. The data synchronicity status for that specific synchronous consumer DSA changes to **TRUE**. When a master DSA transmits an incremental update to synchronous consumer DSAs, it waits for a result from each synchronous consumer DSA whose data synchronicity status is **TRUE**. The master DSA

proceeds differently depending on the result, as shown in the following table.

| Incremental update result | Procedure |
|---|---|
| SUCCESSFUL | No status change. |
| Shadow error (synchronous consumer DSA processed update and returns error) | Change data synchronicity status to FALSE.<br><br>Start error recovery procedure **b)** |
| Network error | Change data synchronicity status to FALSE.<br><br>Start error recovery procedure **b)** |
| TIMEOUT **a)** | Change data synchronicity status to FALSE.<br><br>Continue to wait for result until SUCCESSFUL, shadow error or network error received. |

As indicated in the table:

a. The master DSA monitors the time it spends waiting to receive a synchronous consumer DSA's incremental update result. If a given time interval is exceeded, the master DSA raises a TIMEOUT error. Administrators can configure the maximum waiting time interval with the DirX Directory environment variable **DIRX_DSA_SYNC_TIMEOUT**. The default value is 5 seconds. See the chapter "DirX Directory Environment Variables" in the *DirX Directory Administration Reference* for details.

b. The error recovery procedure starts as defined in the shadow initiator policies. A master DSA typically retries an incremental update until it receives a successful result or the maximum number of retries is reached, at which time it disables the shadowing agreement.

Once the data synchronicity status of synchronous consumer DSA changes to **FALSE** and the shadowing agreement is not disabled, the master DSA tries to propagate all outstanding incremental updates to the synchronous consumer DSA. If no further errors occur, the synchronous consumer DSA's data will catch up with the master DSA after a period of time and its data synchronicity status changes back to **TRUE**. This information is also propagated within the incremental update.

A high data-availability scenario can require at least one synchronous consumer DSA with a data synchronicity status of **TRUE** to be online at any given time for a master to accept update operations; that is, if you have configured two or more synchronous consumer DSAs, the master DSA should reject an update operation (with the error code "unwilling to perform the operation") unless at least one of these consumer DSAs is online and has a data synchronicity status of **TRUE**. Use the DirX Directory environment variable **DIRX_FORCE_SYNC_CONSUMER** to enable this feature. See the chapter "DirX Environment Variables" in the *DirX Directory Administration Reference* for details.

### 9.1.3. Evaluating Asynchronous and Synchronous Shadowing Performance

Synchronous shadowing has a much greater impact on performance than asynchronous shadowing does. Although the performance numbers given in this section are derived from an actual scenario, they serve only for discussion purposes and may not reflect performance numbers in your configuration.

First we'll look at the response time for a typical modify operation measured at the DirX Directory LDAP server. When asynchronous shadowing is enabled (the *disp_type* value in the
**–consumerkind** option is **CENTRALADMIN**; see the **dirxadm sob create** operation in the *DirX Directory Administration Reference* for details), this time may take up to three (3) milliseconds (ms) and includes:

- The time required for decoding and encoding the request and result.
- The time it takes the DSA to perform the update operation in the local database and write the incremental update into the journal.

Independent of asynchronous consumer DSA performance, a subsequent modify operation will still take about three (3) milliseconds, except for some local performance fluctuations due to disk flushes and so on.

When synchronous shadowing is enabled (the *disp_type* value in the **–consumerkind** option is **SYNCHRON**; see the **dirxadm sob create** operation in the *DirX Directory Administration Reference* for details), the response time will be greater since the time for transmitting the incremental update PDU and receiving the result PDU to/from synchronous consumer DSAs is added plus the time for performing the modify operation at synchronous consumer DSA.Thus, response time increases up to six (6) milliseconds.

Another aspect is network latency, which depends on the distance between DSAs, installed network equipment and type of cable, among other items.For example, a fiber optic cable typically brings about 5 microseconds (µs) of latency per kilometer.For longer distances, this value may be even greater due to the deployment of amplifiers or regenerators.In our example, response time may increase up to eight (8) milliseconds (ms) or greater if the cable distance between DSAs is 200 kilometers (km).

Another interesting performance measure is the maximum possible update operations per second within a synchronous shadowing scenario.This number is influenced by many factors, such as system performance, network latency, PDU size and type of modification, and is thus impossible to predict exactly without knowing what these characteristics are.Performance is certainly not the reciprocal of response time, since multiple operations are processed simultaneously.However, you should take into account that the slowest consumer DSA at the greatest distance slows down update operation performance in synchronous shadowing scenario.

## 9.2. Planning the Shadow Configuration

My-Company wants to create a directory service that offers the highest data availability that DirX Directory can provide today.The company operates multiple geographically-

separate data centers in Germany and one data center for its subsidiary in Boston.

After evaluating the issues of performance against the issues of high availability described in “Understanding Asynchronous and Synchronous Shadowing”, My-Company decides to implement three DSAs in the Greater Munich area and one DSA in Boston with the following characteristics:

- A single master DSA (DSA1) and two synchronous consumer DSAs (DSA2 and DSA3) located in the Greater Munich area within 50 kilometers of each other to minimize network latency.These DSAs build My-Company’s directory data back end and use the synchronous shadowing protocol for replication to ensure the highest level of data integrity.These three DSAs carry the main load of DAP and LDAP requests.DSA2 and DSA3 act as accessible DSAs for load distribution and as online backup for the directory data.They can also be switched to master the directory data if necessary.As a result, DSA1, DSA2 and DSA3 all have the same system configuration for best performance.
- One asynchronous DSA (DSA4) located in Boston, Massachusetts to perform DAP and LDAP requests close to the STU subsidiary’s users and applications.This DSA uses asynchronous shadowing protocol due to the great distance between locations.DSA4’s system configuration can differ from the DSAs in Greater Munich to address STU’s requirements.

The following figure illustrates this solution.

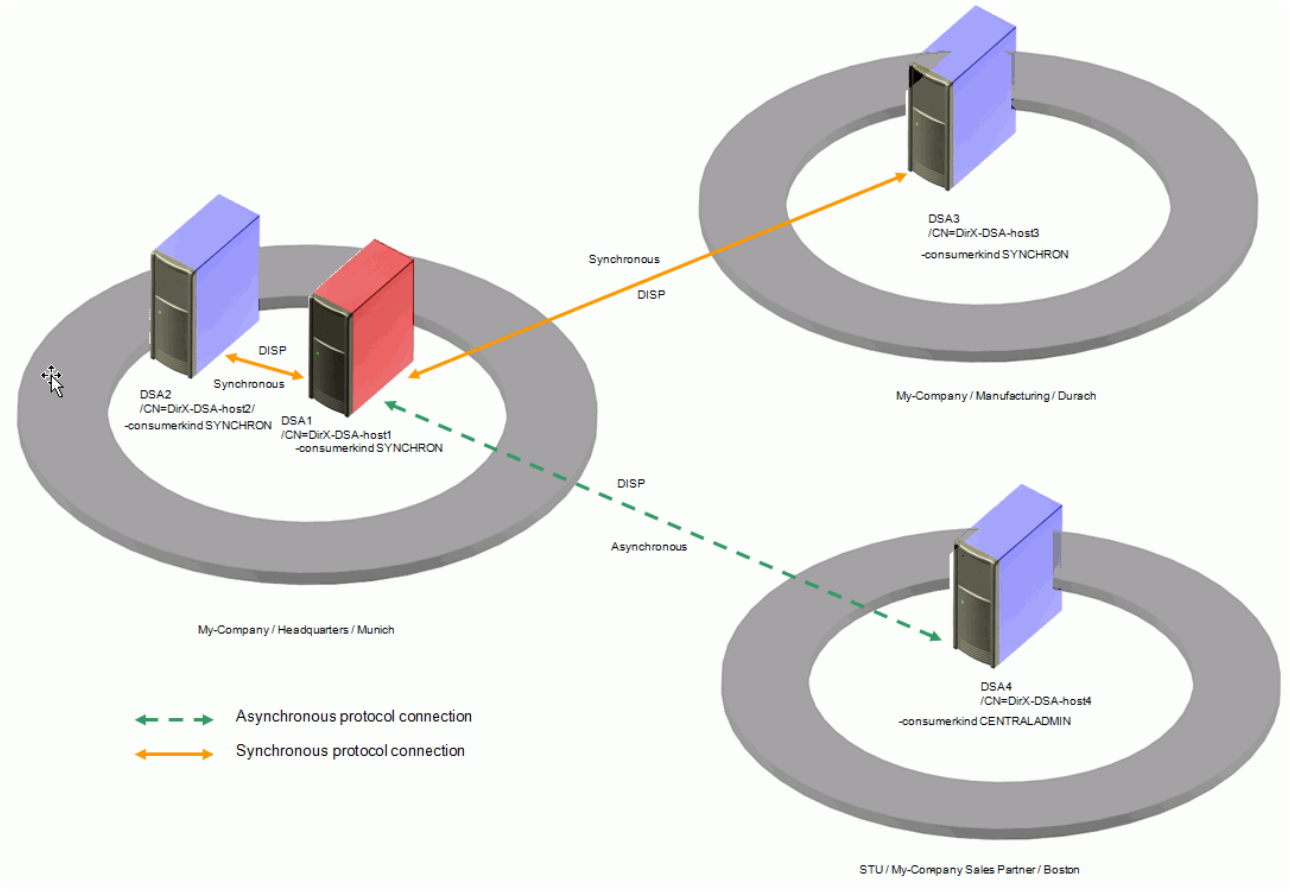



*Figure 33. My-Company Synchronous Shadow Configuration*

# 9.3. Building the Shadow Configuration

To build the configuration, the administrators of DSA1, DSA2, DSA3 and DSA4 first:

- Initialize the supplier DSA1 and populate it with My-Company's directory data.
- Initialize the two synchronous consumer DSAs (DSA2 and DSA3) with **dbamboot** and then start them with empty databases.
- Initialize the one asynchronous consumer DSA (DSA4) with **dbamboot** and then start it with empty database.

The DSA administrators then:

- Create the shadowing agreements for the DSAs
- Configure the attribute indexes for the DSAs

The next sections describe these procedures.

## 9.3.1. Creating the Shadowing Agreements

The DSA1 administrator creates the synchronous shadowing agreements for DSA2 and DSA3 on the supplier DSA1. Since synchronous shadowing is applied to these DSAs, their shadowing agreement configuration is as follows:

- The complete DIT starting at the root DSE (/) and including all object classes and all attributes is replicated.
- The total update can be performed via DISP or by media. For a large number of entries (more than one million), it can be beneficial to perform a total update by media because it may take less time than a total update via DISP. To do so, specify **-agreement "CHANGEO=TRUE"** in the **dirxadm sob create** command.
- Incremental changes are replicated on change. (Synchronous shadowing does not allow scheduled agreements.)
- The automatic total update parameter specified in SOB-Policies is set to **TRUE** (to guarantee unattended data adjustment from supplier DSA to consumer DSA after an emergency switch).
- The **REPLS** subcomponent of the consumer policy is set to **TRUE**.

The following **dirxadm** command creates the synchronous shadowing agreement for DSA2:

```
sob create \
   -supplier {/CN=DirX-DSA-host1} \
   -supplierpsap
"TS=DSA1,NA='TCP/IP_IDM!internet=123.45.67.91+port=21200',DNS='(HOST=
host1,PLAINPORT=21200)'" \
   -consumer {/CN=DirX-DSA-host2} \
```

```
   -consumerpsap
"TS=DSA2,NA='TCP/IP_IDM!internet=123.45.67.92+port=21200',DNS='(HOST=
host2,PLAINPORT=21200)'" \
   -consumerkind SYNCHRON \
   -agreementid 200 \
   -status cooperative \
   -pol "INI={ATU=TRUE,RMR=60,RI=120},SUPP={US=SEGM-TOTAL-
INCR,MAXLOC=512},CONS={REPLS=TRUE}"
```

The DSA1 administrator uses a similar **dirxadm sob create** to create the synchronous shadowing agreement for DSA3, specifying the consumer-specific data for DSA3 and an agreement identifier of **300**.

Meanwhile, the DSA4 administrator in Boston must also create a shadowing agreement. This agreement is an asynchronous shadowing agreement that:

- Replicates the context **/O=My-Company** and all entries below it.
- Sets the **REPLS** subcomponent of the consumer policy to **FALSE**, because DSA4 uses asynchronous shadowing (the **–consumerkind** *disp_type* option is set to **CENTRALADMIN**) and only synchronous consumer DSAs can assume mastership of directory data in mixed synchronous and asynchronous shadowing configurations.

The following **dirxadm** command creates the asynchronous shadowing agreement for DSA4:

```
sob create \
   -supplier {/CN=DirX-DSA-host1} \
   -supplierpsap
"TS=DSA1,NA='TCP/IP_IDM!internet=123.45.67.91+port=21200',DNS='(HOST=
host1,PLAINPORT=21200)'" \
   -consumer {/CN=DirX-DSA-host4} \
   -consumerpsap
"TS=DSA4,NA='TCP/IP_IDM!internet=198.76.54.40+port=21200',DNS='(HOST=
host4,PLAINPORT=21200)'" \
   -consumerkind CENTRALADMIN \
   -agreementid 400 \
   -status cooperative \
   -agreement "SS={AREA={CP={/O=My-Company}, \
                      RA={DEF=TRUE}},ATT={DEF=TRUE}}, \
                      UM={SI={OC=TRUE}}"
   -pol "SUPP={US=SEGM-TOTAL-INCR},CONS={REPLS=FALSE}"
```

### 9.3.2. Performing a Total Update by Media

To perform a total update by media with synchronous shadowing, the DSA1 administrator:

1. Sets up an empty database on the consumer DSA2 and then starts the DirX Directory service.
2. Creates the agreement on the supplier DSA1 and issues a **sob create** command with **-agreement "CHANGEO=TRUE"**. After creating the agreement with status **COOPERATIVE**, the supplier DSA1 sets the update status to **disabled=TRUE**.
3. Uses the **dirxbackup** utility to save the database into an archive. Please consider the following behavior of **dirxbackup**:
    1. Only verified backups can be restored. This restriction avoids propagating database inconsistencies to the consumer DSAs.
    Verification must be performed on the consumer when it has a newer version of DirX Directory.
    2. Database verification can last for hours when the database is huge.
    3. By default, full database verification is triggered automatically after backup has finished (see **dbamverify -AXDST**).
    4. To reduce the timeframe between backup creation on the master and restoration on the consumer, the DSA1 administrator can:
        1. Create the backup with the **-n** option and skip verifications.
        2. Copy the created backup to the consumer.
        3. Verify the backup without the time-consuming attribute index checking using the command:
        **dbamverify -DST** (the A and X options may be omitted to reduce verification time). In this case, the administrator must use the **dirxbackup -n AX** option when restoring the archive.
4. Uses **dirxbackup** on consumer DSA2 to restore the archive. The DirX Directory service restarts automatically after **dirxbackup** has finished.
5. Enables the agreement on supplier DSA1. The consumer DSA2 must be online because the supplier DSA1 checks whether the correct archive was restored on consumer DSA2: the supplier DSA1 has generated a backup signature in the subentry **/CN=DirXDBVersionSubentry** which is compared with the backup signature at consumer DSA2. If the backup signatures are not identical, the enable operation returns an error. (If this verification procedure does not comply with your operational procedures, you can disable it by setting the environment variable **DIRX_DSA_DISP_TUBM** to **0**. This setting prevents backup signature generation and comparison at enable time.)

An alternative method for performing a backup is to initiate an LDIF dump with the **dirxadm ldif_dump** command. In this scenario, the DSA1 administrator:

- Copies the generated LDIF file(s) to the consumer DSA2.
- Stops the consumer DSA2.
- Resets the database with the **dbamboot** command.

- Loads the entries with the **dirxload** command, specifying the **-A** option to perform the attribute index configuration as configured on the supplier DSA1.
  When multiple LDIF files must be loaded, specify all LDIF files in one invocation of the **dirxload** command.
- Starts the DirX Directory service on consumer DSA2.
- Enables the agreement on the supplier DSA1.

You can also perform the total update by media for multiple consumer DSAs in one sequence. In this scenario, the DSA1 administrator:

- Creates the synchronous agreements with total update by media for both consumer DSAs DSA2 and DSA3.
- Saves the database with **dirxbackup**.
- Restores the archive on DSA2 and DSA3.
- Enables the agreements.

The procedure is the same when using an LDIF dump.

To restore an older **dirxbackup** archive, the administrator performs the following steps to synchronize the supplier DSA and the synchronous consumer DSAs:

- Restores the **dirxbackup** archive on supplier DSA1. The DirX Directory service restarts automatically and the supplier DSA1 sets the update status to **disabled=TRUE** for all agreements where **CHANGEO=TRUE** is specified.
- Creates an archive with **dirxbackup** or an LDIF dump.
- Restores the archive or load the LDIF file(s) on the consumer DSAs.
- Enables the agreements.

### 9.3.3. Configuring the Attribute Indexes

Attribute index configurations are not inherited from master to consumer DSAs.As a result, the administrators of DSA1 through DSA4 must individually configure their DSA attribute indexes with the **dirxadm db attrconfig** command (see the **dirxadm** section in the *DirX Directory Administration Reference* for details).In this sample configuration, the Greater Munich administrators configure identical attribute index configurations for DSA1, DSA2 and DSA3 because these DSAs act as equivalent directory servers that handle the same kind of operation requests.The DSA4 administrator creates a different attribute index configuration for DSA4 that matches the workload expected from LDAP applications in Boston.

## 9.4. Extending the DIT with New Company Data

My-Company has purchased a small company to enhance its product portfolio.Because this new company is also to use My-Company's directory service, its directory data needs to be migrated to My-Company's directory.

New-Company's entries are to be loaded below a new context prefix **O=New-**

**Company**.Since synchronous shadowing is in force, the DSA1 administrator must use **dirxcp** to create the new context prefix, because an entry created with **dirxadm** will not be shadowed.

The following **dirxcp** command creates the new context prefix **O=New-Company**:

```
create /O=New-Company -attr  \
   {OCL=TOP;ORG} \
   {DSET=ENTRY+ADM_POINT+CP} \
   {AR=AA;ACSA;CASA} \
   {ACS=SACS} \
   {SACI={ID=admin: enable Handling of Subentries,
         PR=254,
         AL={BL={L=SIMPLE}},
         UF={UC={N={DN={/O=New-Company/cn=admin}};
                              {DN={/O=My-Company/cn=admin}}},
             UP={PI={E=TRUE},
                 GAD=grantDiscloseOnError+grantRead+grantBrowse+
                     grantReturnDN+grantAdd+grantRemove+grantModify};
                 {PI={AT=SS;OC;AT;DSR;NF;DCR;MR;MRU;ATIDX;CRN;CRT;
                         MN;MT;RFCVN;RFCVV;AT;OC;NOS;NOAS;DUUID;
                         ATIDX;PACI;EACI;SACI;AR;ACS;DSET,
                      AAV=SS;OC;AT;DSR;NF;DCR;MR;MRU;ATIDX;CRN;CRT;
                          MN;MT;RFCVN;RFCVV;AT;OC;NOS;NOAS;DUUID;
                          ATIDX;PACI;EACI;SACI;AR;ACS;DSET,
                      AUATV=TRUE},
                  GAD=grantDiscloseOnError+grantRead+grantCompare+
                      grantFilterMatch+grantAdd+grantRemove}}};
         {ID=allow anonymous to read,
          PR=10,
          AL={BL={L=NONE}},
          UF={UC={AU=TRUE},
              UP={PR=0,
                  PI={E=TRUE,AUATV=TRUE,

AT=RFCVN;RFCVV;AT;OC,AAV=RFCVN;RFCVV;AT;OC;ATIDX},
                  GAD=grantDiscloseOnError+grantRead+grantBrowse+
                      grantReturnDN+grantFilterMatch}}}}
```

The DSA1 administrator also creates an administrator entry and an access control-specific subentry for the new context prefix. See the chapters “Setting up the DSA” and Setting up the DirX Directory Service” for descriptions and examples of these tasks.

Next, the DSA1 administrator populates the tree with entries from **/O=New-Company**; for example, with **dirxmodify**, as described in the section “Importing Directory Updates from an LDIF File” in the chapter “Using LDIF Files for Synchronization”. The DSA then automatically replicates all the new entries starting with **/O=New-Company** to DSA2 and DSA3 without the need for a new shadowing agreement, since these DSAs participate in synchronous replication.

The DSA1 administrator, however, must set up a new shadowing agreement to replicate the New-Company tree to DSA4. The administrator uses the **dirxadm sob create** command and specifies a **–consumerkind** *disp_type* value of **CENTRALADMIN** to replicate the **/O=New-Company** context prefix, as follows:

```
sob create \
   -supplier {/CN=DirX-DSA-host1} \
   -supplierpsap
"TS=DSA1,NA='TCP/IP_IDM!internet=123.45.67.91+port=21200',DNS='(HOST=
host1,PLAINPORT=21200)'" \
   -consumer {/CN=DirX-DSA-host4} \
   -consumerpsap
"TS=DSA4,NA='TCP/IP_IDM!internet=198.76.54.40+port=21200',DNS='(HOST=
host4,PLAINPORT=21200)'" \
   -consumerkind CENTRALADMIN \
   -agreementid 410 \
   -status cooperative \
   -agreement "SS={AREA={CP={/O=New-Company}, \
                       RA={DEF=TRUE}},ATT={DEF=TRUE}}, \
                       UM={SI={OC=TRUE}}"
   -pol "SUPP={US=SEGM-TOTAL-INCR},CONS={REPLS=FALSE}"
```

## 9.5. Disabling and Enabling the Shadowing Agreements

The My-Company administrators want to install the latest system and DirX Directory software patches onto the consumer DSAs.The first step in this process is to disable the shadowing agreement for the consumer at the master DSA.Once this agreement is disabled, the master DSA stops sending incremental updates to the consumer DSA, but continues to commit subsequent incremental updates and propagates them to consumer DSA once the agreement is re-enabled.

The following **dirxadm** command disables the shadowing agreement **200** for DSA2:

```
sob disable \
   -supplier {/CN=DirX-DSA-host1} \
```

```
   -consumer {/CN=DirX-DSA-host2} \
   -agreementid 200
```

Now the DSA2 administrator can stop DSA2 and install the system and DirX Directory patches.Once he has completed the installations and re-starts the DirX Directory Service, the administrator at the master DSA (DSA1) can enable agreement **200** for DSA2 with the following **dirxadm** command:

```
sob enable \
   -supplier {/CN=DirX-DSA-host1} \
   -consumer {/CN=DirX-DSA-host2} \
   -agreementid 200
```

## 9.6. Monitoring Data Synchronicity Status

After enabling the shadowing agreement as described in the previous section, DSA2's data may not be synchronized with the data on the master DSA1 because update operations may have been performed while DSA2 was off-line.DirX Directory offers several ways to evaluate data synchronicity in a synchronous shadowing configuration:

- Evaluating the recent modification sequence number (recent MSN) on the master and consumer DSAs
- Evaluating the **/CN=DirXDBVersionSubentry** entry and its attributes
- Evaluating DSA logging, DSA audit and SNMPv2 trap information

The next sections discuss these methods in the context of our scenario.

### 9.6.1. Using Recent MSN Values

Recall from the section "How Data Synchronicity is Managed" that master and consumer DSAs keep a recent modification sequence number (recent MSN) operational attribute (in the entry **/CN=DirXDBVersionSubentry**). One way to determine whether data is synchronized between a master DSA and a synchronous consumer DSA is to compare the DSAs' recent MSNs. Since the master DSA knows exactly what it has sent to the consumer DSA and what the consumer DSA has confirmed, you can easily evaluate the data status with **dirxadm sob show** commands at the master and consumer DSAs.

The following **dirxadm** command run on the master DSA1 shows the value of its recent MSN:

```
dirxadm> sob show -p
Supplier-DSA
Cooperating-DSA
  AE-Title      : /CN=DirX-DSA-host1
```

```
PSAP-Address
  T-Selector    : DSA1
  NSAP-Address  : TCP/IP_IDM!internet=123.45.67.91+port=21200
  DNS           : (HOST=host1,PLAINPORT=21200)
  DISP-Kind     : SYNCHRON
Recent-MSN
  Recent-MSN    : 12346
Sync-Status
  Sync-Status   : TRUE
```

As you can see, the master's Recent-MSN attribute has a value of **12346**. The Sync-Status (this is the DirX Directory In Sync operational attribute described in the section "How Data Synchronicity is Managed") at the master DSA is always **TRUE**.

The following **dirxadm** command run on the master DSA1 shows the values of the consumer DSA recent MSNs:

```
dirxadm> sob show -suppl {/CN=DirX-DSA-host} -p
Consumer-DSA
Cooperating-DSA
  AE-Title      : /CN=DirX-DSA-host2
PSAP-Address
  T-Selector    : DSA2
  NSAP-Address  : TCP/IP_IDM!internet=123.45.67.92+port=21200
  DNS           : (HOST=host2,PLAINPORT=21200)
  DISP-Kind     : SYNCHRON
Recent-MSN
  Recent-MSN    : 9875
Sync-Status
  Sync-Status   : FALSE
Cooperating-DSA
  AE-Title      : /CN=DirX-DSA-host3
PSAP-Address
  T-Selector    : DSA3
  NSAP-Address  : TCP/IP_IDM!internet=123.45.67.93+port=21200
  DNS           : (HOST=host3,PLAINPORT=21200)
  DISP-Kind     : SYNCHRON
Recent-MSN
  Recent-MSN    : 12346
Sync-Status
  Sync-Status   : TRUE
```

```
Cooperating-DSA
  AE-Title      : /CN=DirX-DSA-host4
PSAP-Address
  T-Selector    : DSA4
  NSAP-Address  : TCP/IP_IDM!internet=198.76.54.40+port=21200
  DNS           : (HOST=host4,PLAINPORT=21200)
  DISP-Kind     : CENTRALADMIN
```

As you can see from this output, DSA2's Recent-MSN attribute is **9875**. Its Sync-Status (DirX Directory In Sync attribute) is also **FALSE**, which indicates that DSA2's data is not synchronized with the master DSA because its Recent-MSN is different (and was caused by taking DSA2 off-line for a period of time). DSA3's Recent-MSN value is the same as DSA1, because DSA3 remained online. DSA4 does not show a Recent-MSN value, because it is an asynchronous consumer DSA and thus does not maintain this attribute and the other synchronous shadowing attributes. Subsequent status checks of DSA2's Recent-MSN attribute over time show that it increases and eventually reaches the same value as the master DSA1.

### 9.6.2. Using the DirXDBVersion Subentry

Recall from the section "How Data Synchronicity is Managed" that the entry **/CN=DirXDBVersionSubentry** and its operational attributes are administered by both master and synchronous consumer DSAs. Another way to check a consumer DSA's data synchronicity status is to read this subentry and its attributes using **dirxcp**/DAP or **dirxcp**/LDAP operations. First, make sure you set the service controls that direct the DSA to perform the read operation locally, as shown in the **dirxcp**/DAP example below (in the **dirxcp**/LDAP example, the default LDAP server service control settings have already been considered). For more information about service controls, see the **dirxcp args** and **ldapargs** operations in the *DirX Directory Administration Reference*.

The following **dirxcp**/DAP commands read the entry **/CN=DirXDBVersionSubentry** from DSA2:

```
dirxcp> bind -dsa /CN=DirX-DSA-host2 [your bind arguments]
dirxcp> args modify -localscope true
dirxcp> args modify -subentries true
dirxcp> args modify -copyshalldo true
dirxcp> args modify -dontusecopy false
dirxcp> show /cn=dirxdbversionsubentry -alla  -p
1) /CN=DirXDBVersionSubentry
    From-Entry                : FALSE
    Object-Class              : DBVS
                              : TOP
                              : SUBE
```

```
    Common-Name                : DirXDBVersionSubentry
    dirx-entry-uuid            : 76b74ea7-ef95-47e4-aa51-11dc051a9e0
    Recent-MSN                 : 12346
    Recent-MSN-Time-Stamp      : 20121204085841.987Z
    Recent-Distinguished-Name  : /O=My-Company/OU=ou1/CN=1-12345
    Recent-Operation           : add
    Consumer-In-Sync           : TRUE
```

You can also display this subentry with the following **dirxcp**/LDAP command:

```
dirxcp> set _errormsg TRUE
dirxcp> search  cn=DirXDBVersionSubentry -base -filt ocl=* -vtype
subentry -attr + -p
 1) cn=DirXDBVersionSubentry
    dirxEntryUUID          : 76b74ea7-ef95-47e4-aa51-11dc051a9e0
    dirxRecentMSN          : 12346
    dirxRecentMSNTimeStamp : 20121204085841.987Z
    dirxRecentDN           : cn=1-12345,ou=ou1,o=My-Company
    dirxRecentOperation    : add
    dirxInSync             : TRUE
LDAP-Result: Search succeeded. Found 1 Entries (0 Aliases), 6
Attributes, 6 Values. (ChainedResult=no)
```

In the **dirxcp**/DAP output, the From-Entry attribute shows the value **FALSE**, which indicates that this entry comes from the consumer DSA. In the LDAP example, this fact is indicated by the **(ChainedResult=no**) comment. The DirX Directory Recent DN attribute (LDAP name dirxRecentDN) shows the DN of the last update operation. The DirX Directory Recent Operation attribute (LDAP name dirxRecentOperation) shows the type of request of the last update operation. Valid values are **add**, **remove**, **modify** and **modifydn**. The DirX Directory Recent MSN Time-Stamp attribute (LDAP name dirxRecentMSNTimeStamp) shows the modification time stamp of the last update operation. The DirX Directory Recent MSN attribute (LDAP name dirxRecentMSN) shows the modification sequence number of the last update operation. The DirX Directory In Sync attribute (LDAP name dirxInSync) of the consumer DSA shows the status of data synchronicity. The value is **TRUE**, which indicates that DSA2's data is synchronized with the master DSA's data.

### 9.6.3. Using Logging, Audit and SNMPv2 Information

Recall from the section "How Data Synchronicity Status Affects Read Operation Service Controls" that synchronous shadowing's data synchronicity management overrides service controls for read and search operations in favor of accurate results.

In our example, DSA2's data is not yet synchronized after the upgrade procedure, but DSA2 can perform read and search operations when it comes online. As long as DSA2's data is

out of sync with the master DSA1, it chains the read/search requests to the master DSA1. You can follow the changes in a consumer DSA's synchronicity status with:

- DSA logging at the master DSA and the synchronous consumer DSA
- DSA auditing at the master DSA and synchronous consumer DSA, if audit is enabled
- SNMPv2 traps generated at the master DSA and the synchronous consumer DSA, if SNMPv2 traps are enabled.

When a consumer DSA becomes unsynchronized with the master DSA, the following notice is reported in DSA logging:

```
0 Operational Binding 200 switches to "asynchronous" state
Reason: clear text message
```

When a consumer DSA becomes synchronized with the master DSA, the following notice is reported in DSA logging:

```
0 Operational Binding 200 switches to "synchronous" state
Reason: clear text message
```

As you can see, the notices refer to an operational binding ID that is synonymous with the shadow agreement ID.The *clear text message* string gives the reason for the status change along with a simple explanation.

Synchronicity status changes are also reported in DSA audit records and via SNMPv2 traps, if these DirX Directory monitoring tools are enabled.See the chapter "Monitoring DirX Directory" for more information about these tools.

# 9.7. Switching Supplier DSAs

This section describes how to switch supplier DSAs operating in a synchronous shadowing configuration in both non-emergency and emergency situations.

## 9.7.1. Making a Non-Emergency Switch

Switching supplier DSAs in a synchronous shadow configuration functions in much the same way as in an asynchronous shadow configuration: a non-emergency switch is performed when all of the DSAs participating in the synchronous shadow configuration are online and their agreements are not disabled.The synchronous master DSA propagates all outstanding incremental updates to all synchronous and asynchronous consumer DSAs.This procedure can take some time; for example, when a scheduled shadowing agreement to an asynchronous consumer DSA exists.

As an alternative, you can perform a **sob synchronize** command to propagate all outstanding incremental updates before you invoke the **sob switch** command.After all outstanding incremental updates have been propagated, the master DSA updates all

shadowing agreements in the cooperating DSA table locally and then propagates the updated table to all consumer DSAs.As long as the non-emergency switch lasts, all update operations are rejected.

In our example, the administrator of master DSA1 needs to upgrade the DSA with the newest system and DirX Directory patches after the consumer DSAs have been upgraded.To provide an uninterruptible DirX Directory service while DSA1 is being upgraded, the administrator performs the following tasks:

- Switches the mastership to one of the synchronous consumer DSAs - in this case, DSA2 - with the following **dirxadm** command run on DSA1:

```
sob switch -from /CN=DirX-DSA-host1 \
           -to /CN=DirX-DSA-host2
```

- Disables the agreement to the old supplier DSA1 with the following command run on the new supplier DSA2:

```
sob disable -supplier /CN=DirX-DSA-host2 \
 -consumer /CN=DirX-DSA-host1 \
 -agreementid 200
```

- Stops DSA1, installs the patches and then starts DSA1 again.
- Enables the agreement to DSA1 with the following **dirxadm** command run on the new supplier DSA2:

```
sob enable -supplier /CN=DirX-DSA-host2 \
 -consumer /CN=DirX-DSA-host1 \
 -agreementid 200
```

- Waits until DSA1's data is in sync with the new supplier DSA2 by following its data synchronicity status as described in "Monitoring Data Synchronicity Status". (You can omit this step if you want.)
- Switches mastership back to DSA1 with the following **dirxadm** command run on DSA2:

```
sob switch -from /CN=DirX-DSA-host2 \
           -to /CN=DirX-DSA-host1
```

### 9.7.2. Making an Emergency Switch

A supplier DSA may fail for various reasons. An administrator may perform an emergency switch, depending on the reason for the failure. A crash of the dirxdsa or dirxldapv3 process

does not require an emergency switch since a restart of the DirX Directory service typically lasts within the range of seconds. But a system crash or a network interruption may take a longer time and an emergency switch is recommended. In this case, the administrator can run a **dirxadm sob switch –emergency** command on one of the synchronous consumer DSAs. This command performs the following actions:

- Sets the DSA to read only.
- Tests network connectivity to the supplier DSA. If this test succeeds at connecting to the supplier DSA, the operation rejects the emergency switch.
- Determines the highest recent modification sequence number of all synchronous consumer DSAs and rejects the emergency switch if the performing consumer DSA doesn't own this value. That is, the performing consumer DSA contacts each synchronous consumer DSA and reads the dirxRecentMSN attribute. These values are compared with the value of the local dirxRecentMSN attribute. If the local value is exceeded by a value from any other synchronous consumer DSA, the emergency switch is rejected.
- Updates all shadowing agreements in the cooperating DSA table with the new supplier DSA.
- Propagates the cooperating DSA table to all consumer DSAs.
- Releases read only state.

Once the old supplier DSA is online again, the new supplier DSA sends a total update if the maximum recovery time has not been exceeded. Please refer to the section "SOB-Policies" in the *DirX Directory Syntaxes and Attributes* for more information on this topic. If the maximum recovery time has been exceeded, the shadowing agreement to the old supplier DSA is disabled and the administrator should be enable it manually,

Under normal conditions, all synchronous consumer DSAs should have equal recent modification sequence numbers at the time of the emergency switch. However, it is possible that a synchronous consumer DSA's data may be out of sync, as indicated by a difference in the recent modification sequence number when compared to the new supplier DSA. In this case, the supplier DSA automatically sends a total update to the relevant synchronous consumer DSA after the emergency switch.

Returning to our example, suppose that the master DSA1's system crashes and that synchronous consumer DSA2's data is still not in sync (DSA2's dirxRecentMSN attribute should have a value of **9999**, for example). An emergency switch at DSA2 is rejected, as shown in the following example:

```
dirxadm> bind -h host2
dirxadm> sob switch -emergency
Error: State of Agreement does not allow the operation to be
performed.
       (Consumer for Binding ID: 300 has higher RMSN)
```

On synchronous consumer DSA2, the following actions are performed:

- A bind to master DSA1 fails since the system has crashed and the system is not online again. The first condition is met and the emergency switch continues.
- DSA2 binds to DSA3 and reads the dirxRecentMSN attribute, which has a value of **12346**. By comparing this value to its own value, DSA2 detects that DSA3 has a higher recent modification sequence number. The emergency switch is rejected and **dirxadm** reports the shadow agreement identifier of the synchronous DSA that owns the highest recent modification number. The administrator needs to initiate an emergency switch manually at DSA3.

After the emergency switch has completed, the new synchronous supplier DSA sends a total update to all synchronous consumer DSAs whose recent modification sequence number was different from the new supplier DSA's at the time of the switch. In our example, DSA3 sends a total update to DSA2, since DSA2's recent modification sequence number had a lesser value than that of DSA3.

The agreement to the old supplier DSA1 goes into the recovery state: the new supplier DSA3 tries to connect to DSA1 every 2 minutes. This time interval is specified in the **RI=*_recovery_interval_ component of the SOB-Policies attribute. If DSA2 can't be connected to within the maximum number of retries (component *RMR**), then DSA3 sets the agreement to the disabled state and won't try any further connections. In this case, the administrator needs to enable the shadow agreement manually once DSA1 is online again.

Since the recent modification sequence number is not maintained at asynchronous consumer DSAs, the new supplier DSA can't evaluate the data synchronicity status at emergency switch time. Thus, it is necessary to invoke a total update to asynchronous consumer DSAs manually to be sure that asynchronous consumer DSAs do not miss any updates due to the system crash on DSA1. A typical method to initiate a total update is to terminate and then establish an asynchronous shadowing agreement. If the asynchronous shadowing agreement is configured for total update by media, perform the following procedures:

- Terminate all agreements configured to the asynchronous consumer DSA.
- Establish all agreements configured to the asynchronous consumer DSA.
- Save the database with **dirxbackup** at the supplier DSA.
- Stop the DirX Directory service at the consumer DSA, restore the database with **dirxbackup**, and then start the DirX Directory service again.
- Enable all agreements configured to the asynchronous consumer DSA.

In this example, the shadowing agreements to asynchronous consumer DSA4 are configured to perform a total update by protocol. Consequently, the administrator needs to terminate and establish agreements 400 and 410, as shown in the following **dirxadm** commands:

```
sob terminate -agreementid 400 \
                    -supplier /CN=DirX-DSA-host3 \
                    -consumer /CN=DirX-DSA-host4 \
sob terminate -agreementid 410 \
```

```
                    -supplier /CN=DirX-DSA-host3 \
                    -consumer /CN=DirX-DSA-host4 \
sob establish -agreementid 400 \
                    -supplier /CN=DirX-DSA-host3 \
                    -consumer /CN=DirX-DSA-host4 \
sob establish -agreementid 410 \
                    -supplier /CN=DirX-DSA-host3 \
                    -consumer /CN=DirX-DSA-host4 \
```

# 10. Using Multiple Contact DSAs

This chapter provides information on how to set up and manage a floating master-shadow configuration that distributes DAP processing requests from LDAP servers across the shadow DSAs in the configuration to provide for better DAP load balancing.

The sample shadow configuration provided in this chapter builds on the sample scenarios described in the chapters “Creating a Shadow DSA”, “Distributing the DIT Across Multiple DSAs”, “Multireplication” and “Creating a Synchronous Shadow DSA”. We suggest you read these chapters to familiarize yourself with shadowing, distribution and replication concepts and procedures before reading this chapter.

## 10.1. Understanding the Multiple Contact DSA Configuration

The LDAP servers configured in the DirX Directory service scenarios described in the previous chapters in this guide all use a single contact DSA – a DSA whose name and PSAP address is configured in the LDAP server’s **dirxldap.cfg** file - for performing DAP operations derived from incoming LDAP operations.In these scenarios, there is a one-to-one relationship between the LDAP server and its contact DSA, which means that all LDAP traffic is forwarded to this DSA for processing.

In master-shadow configurations where the complete DIT is replicated to all consumer DSAs and all consumer DSAs have the same system configuration, the consumer DSAs are idempotent: an LDAP request can be forwarded to any one of the consumer DSAs for equivalent processing.This kind of configuration allows for the distribution of LDAP traffic and subsequent DAP operations among the consumer DSAs rather than isolating DAP processing to the DSA connected to the LDAP server and potentially overloading it while the other consumer DSAs remain mostly idle.

To improve DAP load balancing for this kind of master-shadow configuration, administrators can define multiple contact DSAs in an LDAP server’s configuration file.On each new DAP bind, the LDAP server will contact these DSAs on demand using round-robin selection.The following figure illustrates the difference in DAP load balancing between single and multi-contact DSA configurations:

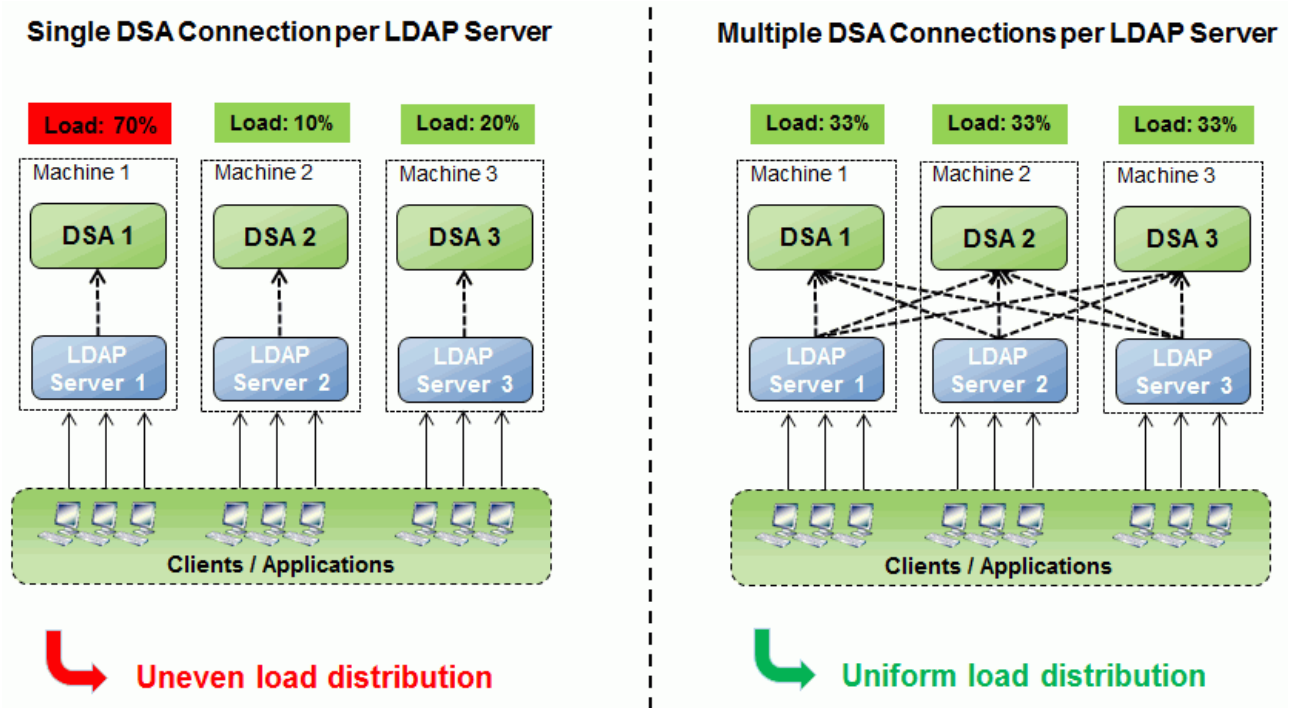



*Figure 34. Single vs. Multiple Contact DSA Configurations*

As shown in the figure, the multiple-contact DSA configuration distributes the load across all DSAs no matter how much load occurs at any of the LDAP server front ends.

### 10.1.1. Configuration Requirements

The multiple-contact DSA configuration has some requirements on DSA, LDAP server and network configuration:

- All contact DSAs must be identical: they must shadow the complete DIT and they must all have the same system configuration.
- The master DSA should not be included as a contact DSA because unexpected operations can occur.For example, unexpected chaining operations can occur when the master DSA is switched.
- The shadow DSAs must use synchronous shadow agreements if the environment supports “read after write” clients and these clients are allowed to contact shadow DSAs.If asynchronous mode is used, there is a race condition between the time it takes for the DISP update from the master to the shadows and how quickly the client performs the read after completing the modify.If the client read occurs before the shadow DSA is updated, the read data will be the old data, not the latest content.
- Network communication must be possible between all contact DSAs and the LDAP server.Consequently, all firewalls and routers in the configuration must permit DAP connections (IDM on top of TCP/IP with port 21200) from all consumer DSA machines to all other consumer DSA machines.
- All contact DSAs should always be available.DSAs that are expected to be frequently off-line—especially during LDAP server startup—should not be included as contact DSAs.However, DSAs that may fail or be taken off-line intermittently can be included without causing problems.

- Contact DSAs must all have similar response times for seamless interaction with LDAP clients.DSAs whose response times are known to be significantly slow —because of geographical distance, number of hops, or slow lines, for example—should not be included as contact DSAs.
- The primary LDAP servers running on the shadow DSAs in the configuration should not be scheduled to re-start at the same time, and each LDAP server should be configured to iterate through its contact DSAs in a different order from the other LDAP servers.Using these techniques optimizes LDAP server startup performance and avoids potential delays in LDAP server startup should one or more DSAs become unresponsive.

## 10.2. Additional Features, Limitations and Issues

The multiple contact DSA configuration offers some benefits to a DirX Directory service installation as well as presenting some limitations, tradeoffs and issues regarding other components in the installation:

- The LDAP server in a multiple contact DSA configuration automatically moves to the next available DSA when it detects that a contact DSA is down and then retries the operation.As a result, this kind of configuration provides a simple failover mechanism as opposed to the single contact DSA configuration.The section titled “Automatic Failover Handling” provides more detail about DSA failover and the consequences of DSA outages in the multiple contact DSA configuration.
- Multi-contact DSA load distribution is independent of external load balancing across LDAP servers; for example, through an external hardware load balancer that distributes the LDAP load across the LDAP servers.If the DirX Directory environment includes an LDAP server load balancer, implementing a multiple-contact DSA configuration is not necessary because the LDAP load balancer provides the same solution, and having multiple contact DSAs does not provide any additional benefit for load balancing.For failover, however, there are real benefits: the LDAP servers will retry a running operation and possibly succeed with the next DSA, keeping the error invisible to the client.This is not the case with load balancers in a single contact DSA environment: they return the DSA failure to the client which then (possibly) retries, prompting the hardware load balancer to select another LDAP server.
- Modifications performed at a shadow DSA in a synchronous shadowing configuration take longer than they do in an asynchronous shadowing configuration because all shadow DSAs must be synchronized before the modification is returned to the LDAP client.You can read more about the effects of synchronous and asynchronous shadowing in the chapter “Creating a Synchronous Shadow”.
- LDAP clients that perform a lot of modifications to the DIT and that connect to shadow DSAs may experience performance degradation because the shadow DSA must forward the operation to the master DSA, which is unnecessary if the client is directly connected to the master.However, the master DSA will also experience heavy load in this scenario regardless of the contact DSA configuration, because the DirX Directory service permits only one DSA to perform updates.So, there is a tradeoff between reducing the search load at the master DSA by distributing it to the shadows and incurring the additional DSP call between the shadow and the master for every modify operation that enters a shadow.Predicting which strategy performs better is difficult and depends on many different parameters, so enabling DAP distribution may not

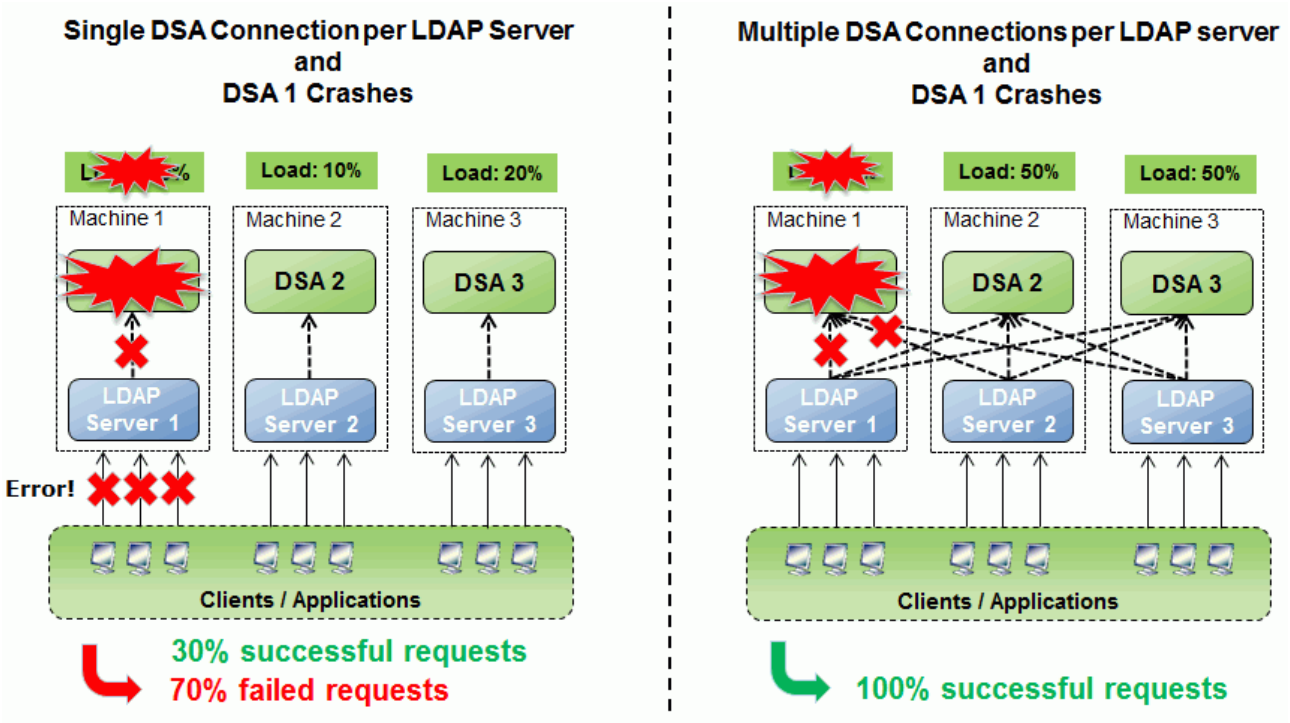
always be the best choice.

- There can be a mix of single-contact and multiple-contact DSA configurations within a set of multiple LDAP servers on a machine (via a subentry-specific LDAP configuration file).If Nagios monitoring is used to query and observe data from a dedicated DSA, the LDAP server that handles the DSA-specific Nagios queries should use the single-contact DSA configuration (specifying the dedicated DSA as its contact) to ensure that the results always come from this DSA.The section “Mixing Single and Multiple Contact DSAs” in this chapter provides an example.

### 10.2.1. Automatic Failover Handling

Although the multiple-contact DSA configuration is intended primarily for DAP distribution, it provides a simple but powerful failover handling capability. With a single contact DSA, the client receives an error when the DSA is down. In a multiple-contact DSA configuration, the LDAP server moves to the next available DSA after detecting the dropout and then retries the operation, allowing the LDAP the client operation to succeed in most cases.

*Single vs. Multiple Contact DSA Configurations when a DSA Fails*




100% success is an idealistic assumption, as there are rare situations in which an error can be returned if the DSA outage occurs at certain stages of an operation’s processing, especially if the operation is running in the DSA while the DSA crashes. In these situations, it depends on exactly where the operation is when the crash occurs. For example, if the operation is returning its result by sending the DAP result PDU back to the LDAP server, the DSA dropout can be experienced at this stage as an invalid PDU received on the LDAP side and is thus handled differently than if just the socket is detected as closed due to the DSA crash. In the long term, most of the situations will end up with a normal socket-close detection on the

LDAP side. In these cases, the LDAP server will retry the operation against another selectable DSA, and the client will see nearly100% success for most cases. However, when severe network problems occur and no DSA can be reached, errors will appear.

#### 10.2.1.1. How the LDAP Server Temporarily Disables a Failing DSA

When the LDAP server detects an unavailable DSA, it automatically disables it from contact DSA selection for a specified time period to avoid frequently repeated retries and errors for this DSA. The LDAP server distributes the requests among the remaining DSAs without any further contact with the DSA it has disabled. When the disable period expires, the DSA becomes selectable again and the LDAP server re-tries it. Depending on the outcome of the re-try, the DSA either remains enabled or is disabled again for the same time period.

The **DIRX_LDAP_AUTO_DISABLE_FAILING_DSA** environment variable controls how long the LDAP server excludes a DSA from selection; the format is

**DIRX_LDA_AUTO_DISABLE_FAILING_DSA**=*nn*

where *nn* is the number of seconds for which the DSA is to be excluded from selection. The default value set in this environment variable is 60 seconds. You can change this default by setting the **DIRX_LDAP_AUTO_DISABLE_FAILING_DSA** environment variable to a different value. See the *DirX Directory Administration Reference* chapter on DirX Directory environment variables for details about this and other DirX Directory environment variables.

#### 10.2.1.2. How the Watchdog Timer Handles DSA Failures

In DirX Directory, the main server processes are started and observed by their watchdog (on Windows, **dirxsrv**; on Linux, **dirxdsas**). The watchdog in earlier versions of DirX Directory runs in a mode where it automatically re-starts the LDAP server when its local DSA crashes, resulting in an unconditional loss of all existing LDAP connections at the time of the DSA crash. In newer DirX Directory versions (V8.5 and higher), the watchdog does not automatically re-start the LDAP server if the local DSA crashes. The LDAP server detects a DSA crash by the drop of the backend DAP connections, and, if running in multiple contact DSA mode, tries to establish new DAP connections to the other contact DSAs and retries the operation. As a result, the DSA outage remains invisible to the client.

Trying a dropped DSA may take some time (due to TCP rules). Consequently, the operation, while ultimately successful, may experience a longer duration than it would have if the DSA was up.

You can enable the previous watchdog mode by setting the **DIRX_WDOG_RESTART_LDAP_ON_DSA_RESTART** environment variable. See the *DirX Directory Administration Reference* chapter on DirX Directory environment variables for more information about this and other DirX Directory environment variables.

#### 10.2.1.3. DSA Outage Handling and its Consequences

If multiple contact DSA are configured, it is necessary to understand the consequences of the fact that a local LDAP server has connections to other DSAs rather than just the local one. As the selection is simply based upon round-robin, a DSA that is permanently down or for a long time will cause every *n*th backend connection to fail. This leads to some consequences. A dropout of a contact DSA can occur at three special times, as seen from the LDAP client's point of view:

- The selected DSA is down and the LDAP client performs a new bind
- The LDAP client has established a connection but no operation is running when the DSA drops out
- The LDAP client has established a connection and the DSA drops out while the client is performing an operation

To understand the reaction to each of these three cases, it is necessary to understand that each LDAP frontend connection is associated with one DAP backend connection to a DSA represented by a DAP handle. The handle is created during LDAP bind time and is stored along with the LDAP connection. Thus, whenever a LDAP operation is performed on this LDAP connection, this DAP handle is used to perform the corresponding DAP request. Now let's examine the consequences for each case.

##### 10.2.1.3.1. Case 1: Selected DSA is Down, Client Performs a New Bind

In this case, a DAP handle does not exist at LDAP bind time. A DAP bind call will fail and no DAP handle is generated. The LDAP server checks if more than one contact DSA is configured and then retries the DAP bind implicitly by selecting the next available DSA until success or until all DSAs have been tried.

If the DAP bind can be established to any of the configured DSAs, the client will receive success, and the dropout of a DSA is invisible.

If DSAs are down when the DAP bind occurs, it may take some time (mainly because of the TCP retry timeout) for TCP to report the unavailability. The time it takes for TCP to detect that a *connect()* call will not be answered depends on many factors and can vary from instantaneously to up to 20-30 seconds depending on the topology and the machine states. For example, connects to machines that are down may experience the full 20-30 second timeout while connects to co-located DSAs are detected instantly.

**Bind Delay due to TCP Timeout**

A DAP bind always starts by establishing a new TCP connection via a *connect()* call. If the target DSA is not reachable (for example, because its host is down), TCP internally retries the *connect()* call using TCP retransmission protocol, which retries the connection establishment several times (configured with TCP parameters) and returns an error only after the last try fails.

This operation leads to the delayed detection of an unreachable peer and results in a delay before the next DSA is selected that cannot be avoided. As a result, even if the next selection finally succeeds, there can be a time delay for the LDAP client before a DAP bind

can be established. Therefore, it is important to configure only DSAs for the selection that are assumed to be “always” available.

be careful about changing TCP parameters to shorten the timeout sequences, as they might have positive effects for the DAP rebind but may have fatal consequences for other aspects. The better choice is to remove “unreachable” DSAs from the contact-DSA list entirely and/or disable them with the **dirxextop** extended operation **ldap_disable_config_dsa**. For details, see the **dirxextop** reference page in the *DirX Directory Administration Reference*.

**Effect of Unexpected Bind Delay**

Due to the possible TCP delay for detecting unreachable peers, an unexpected bind behavior can occur. Let’s assume the LDAP server has two DSAs configured as possible contact DSAs. Let’s further assume the host of DSA1 is down and TCP takes 10 seconds to detect the outage while DSA2 is up and will respond immediately when contacted. Let’s also assume that DSA1 is the one to be selected next for a new DAP Bind.

To better illustrate the effect, let’s also assume that a failing DSA is temporarily (automatic) disabled for only a very short time (less time than it takes for TCP to detect it).

When looking at this scenario, you would assume that 100% of the binds succeed but 50% of the new LDAP binds respond immediately and 50% respond with a delay of 10 seconds. However, this is not the case. To understand why, it is necessary to understand a little how TCP works:

- If an LDAP client performs a new bind, the DAP bind is tried against DSA1. After 10 seconds, the error shows up from TCP and DSA2 is selected. Because DSA2 responds instantly, the LDAP client sees the response after 10 seconds.
- But what about the next new client bind? To which DSA will this bind try the DAP connection? As the previous one succeeded with DSA2, the round-robin algorithm again chooses DSA1 for the next DAP bind and again the TCP delay occurs. Thus, in a scenario where one of the two DSAs is permanently down, *all* LDAP binds see the TCP delay, not just 50% of them.

This behavior is a direct consequence of the simple round-robin algorithm and the fact that 50% of the resources are down. Thus, it is essential to have all configured DSA available all the time in a multiple contact DSA configuration. Usually there are two types of DSA dropouts: accidental and intentional. Intentional dropouts are mostly caused by a machine reboot or by an intentional shutdown of the DirX Directory service—for example, for maintenance—and usually can take some time. For these cases, we recommend using **dirxextop ldap_disable_config_dsa** to disable the corresponding DSA from the selection list of the LDAP servers dynamically before shutting it down or rebooting the host. For details, see the **dirxextop** reference page in the *DirX Directory Administration Reference*.

Accidental dropouts are usually less critical, as TCP usually can detect the missing peer port listener quite rapidly, so no significant delay is to be expected for these cases.

This example shows that you should not set the

**DIRX_LDAP_AUTO_DISABLE_FAILING_DSA** environment variable too low. At a minimum, it should be higher than the TCP retry transmission sequence time for the given host (this is a function of local TCP configuration and typically lies in the range of 10 to 20 seconds).Thus the default of 60 seconds is a good choice in most cases.

#### 10.2.1.3.2. Case 2: LDAP Connection Exists, No Client Operation is Running

An existing LDAP connection always has a corresponding DAP bind represented by the DAP handle. A dropout of the DSA can only be detected when the DAP handle is used again in order to perform the next DAP operation. In this case, the next DAP operation experiences a DAP error indicating that the DAP handle is no longer valid. The LDAP server then tries to re-bind the DAP connection, implicitly iterating through the contact DSA list. On a successful DAP re-bind, the LDAP server exchanges the old (invalid) handle with the new one and re-tries the client operation on the new DAP connection. Thus the DSA dropout is transparent.

### DAP Connection Loss and Paged Searches

If the next operation performed when the DAP handle has meanwhile become invalid is a *nextpage* request of a previously started paged-search request, no re-bind is performed; instead, an LDAP_OPERATION_ERROR is returned. This behavior is necessary because a paged search usually leaves continuation information in the contact DSA that stores information about where to continue when the next page is requested. As this *resume-info* is stored in the memory of the DSA that has received the first page request, no other DSA has any knowledge of it, and so a *nextpage* request cannot implicitly be redirected to another DSA. Instead, the *nextpage* request produces an error and the search cannot continue.

If a *nextpage* operation is performed after a different previous operation was processed, the invalid DAP handle may be restored via rebind in the context of the previous operation and so the *nextpage* will possibly use the wrong DSA, as the rebind may have selected a different DSA than the one that holds the *resume-info* for the *nextpage* call. In this case, the operation returns with an UNWILLING_TO_PERFORM error and an error message indicating that the QueryRef (the paging cookie) is invalid. (This action may only occur if backend sharing is enabled in the LDAP server's configuration.)

#### 10.2.1.3.3. Case 3: LDAP Connection Exists, Client Operation is Running

If the DSA drops out while an operation is being performed, the corresponding DAP operation may experience an internal DAP error (for example, a REMOTE_ABORT, LOCAL_ABORT or BAD_SESSION error). The LDAP server analyzes the error. If a dropout is indicated, the server tries to re-bind the DAP connection (implicitly iterating through the contact DSAs) and on success, re-issues the DAP operation, making the DSA dropout transparent to the client.

Consequently, as long as only one DSA from the configured list is down, the LDAP client should not recognize the DSA outage. However, if more than one DSA is down, LDAP errors may appear even though the co-located DSA is still up and running.

#### 10.2.1.4. How the LDAP Server Handles DSA Outages at Startup

When an LDAP server starts, it tries to read its configuration and schema from the DSA. There is no guarantee that the contacted DSA is the co-located DSA. If the selected DSA is down, the LDAP server tries to come up by re-trying all contact DSAs and continuing if at least one of them is up and returns the required data for configuration.

The LDAP server also establishes a pool of anonymous DAP connections each of which uses a different DSA. If a DSA cannot be contacted, The LDAP server tries the next one until the full DAP pool is established or all DSA are down, in which case the server exits.

##### 10.2.1.4.1. TCP Timeout when Connecting to Dropped-out DSAs

Care must be taken if DSAs are down during LDAP server startup, as the detection of whether a DSA can be reached is completely based upon TCP rules and possibly firewall configuration. If no firewalls are active, a client that tries to connect to a running machine where no DSA is up will receive an immediate reject by the TCP system. In such cases, the LDAP server immediately continues to contact the next DSA. If firewalls or virtual machines are in place, chances are (depending on the set of rules) that they are configured so that rejects are not allowed to be sent out back to the caller, and therefore detection occurs via TCP retry timeouts, which can take up to several seconds (typically 10-20 seconds) before the LDAP server receives a notice of failure from TCP for its *connect()* request. In these cases, the LDAP server startup process may experience a significant delay (*n*-1 * *TCP timeout* in the worst case) before the server finally starts.

Having many DSAs configured but only a few of them up during LDAP startup may lead to minutes of startup time.Consequently, don't configure too many contact DSAs that are likely to be down frequently, especially during LDAP startup.If this situation occurs, either remove these DSAs from the contact DSA list for the LDAP server or—if the server is already up and should not be re-started—disable the DSA for selection using **dirxextop ldap_disable_config_dsa**.For details, see the **dirxextop** reference page in the *DirX Directory Administration Reference*.

#### 10.2.1.5. How Backend Sharing Affects DSA Selection

When backend sharing is active (the default), a new LDAP bind does not necessarily Include the creation of a new DAP backend connection.Therefore, once a DAP bind exists, no new DSA selection takes place if the same user (but a different client) performs another LDAP bind.If a subsequent LDAP operation is forwarded to the DSA, the outage is detected and the DAP connection is restored internally.If backend sharing is active, the restoration may have occurred through some other client meanwhile because both clients share the same DAP connection and the first operation (whoever is the client) on this connection detects the error and causes the internal re-bind.

Consequently, client "A", which starts communicating with DSA1, may perform the next operation against DSA2, although it has never experienced any loss of LDAP connection or received an error.This is the reason why it's important for all DSAs to be idempotent.

If backend sharing is inactive, each new LDAP bind selects the next DSA from the configured list separately.

## 10.3. Planning the Multiple Contact DSA Configuration

Recall from “Creating a Synchronous Shadow DSA” that My-Company administrators set up a floating-master shadow configuration in which three DSAs forming the company’s data center back end use the synchronous shadowing protocol and are located within close geographical proximity to minimize network latency.

After evaluating the benefits, requirements and issues described in “Understanding the Multiple Contact DSA Configuration”, the administrators of the three DSAs in this master-shadow configuration decide to extend the LDAP servers of the two consumer DSAs to support the multiple contact DSA configuration for the following reasons:

- The two consumer DSAs have already been configured to be identical: they shadow the complete DIT and support the same system configuration and they already support the synchronous shadowing protocol.
- The DSAs are in proximity geographically so that they can provide similar response times for optimum interaction with LDAP clients.
- All the firewalls in the configuration permit DAP connections (IDM on top of TCP/IP with port 21200) between DSA2 and DSA3.

## 10.4. Building the Multiple Contact DSA Configuration

Building the multiple contact DSA configuration is a simple task: For each LDAP server that supports one of the consumer DSAs, the administrators need only to remove the master DSA1 as the contact DSA and then add the name and presentation address of the two consumer DSAs to the LDAP server configuration file **dirxldap.cfg** file.

Consequently, the DSA2 administrator updates the DSA contact information in the **dixldap.cfg** file for the DSA2 LDAP server as follows:

```
"/CN=DirX-DSA-host2"
"TS=DSA2,NA='TCP/IP_IDM!internet=123.45.67.92+port=21200',DNS='(HOST=
host2,PLAINPORT=21200)'"
"/CN=DirX-dsa-host3"
"TS=DSA3,NA='TCP/IP_IDM!internet=123.45.67.93+port=21200',DNS='(HOST=
host3,PLAINPORT=21200)'"
```

The DSA3 administrator updates the DSA contact information for the DSA3 LDAP server as follows:

```
"/CN=DirX-DSA-host3"
"TS=DSA3,NA='TCP/IP_IDM!internet=123.45.67.93+port=21200',DNS='(HOST=
```

```
host3,PLAINPORT=21200)'"
"/CN=DirX-dsa-host2"
"TS=DSA2,NA='TCP/IP_IDM!internet=123.45.67.92+port=21200',DNS='(HOST=
host2,PLAINPORT=21200)'"
```

Note that the order in which each contact DSA's information is listed should be different for each **dirxldap.cfg** file.

The administrators then re-start the LDAP servers for the updates to take effect.The multiple contact DSA configuration is now active.

## 10.5. Disabling and Enabling Contact DSAs

Recall from the section "Understanding the Multiple Contact DSA Configuration" that all contact DSAs must always be available to avoid TCP-related bind delays.The administrator for DSA2 determines that its host machine needs to be taken offline for software updates.Since contact DSAs must be removed from a multiple contact DSA configuration before they become unreachable, the DSA2 and DSA3 administrators need to use the **dirxextop ldap_disable_ldap_disable_config_dsa** extended operation to remove DSA2 from the contact DSA configuration information in each LDAP server's **dirxldap.cfg** file.

Administrators of DSA2 and DSA3 both use the following **dirxextop** command to disable DSA2 from their respective LDAP server configuration:

```
dirxextop -D cn=admin,o=my-company -w dirx -t ldap_disable_config_dsa -P
/CN=DSA2
```

Now DSA2 is blocked from being selected as a contact DSA by each LDAP server until the administrators explicitly re-enable it with the **dirxextop** extended operation **ldap_enable_config_dsa** or until the LDAP server is re-started.

Once the update is complete and DSA2 is back online, the administrators use **dirxextop** to re-enable DSA2 as a contact DSA for their LDAP servers with the following command:

```
dirxextop -D cn=admin,o=my-company -w dirx -t ldap_enable_config_dsa -P
/CN=DSA2
```

## 10.6. Monitoring a Multiple Contact DSA Configuration

As multiple DSA selection is performed completely internally and without any impact on the actual LDAP operations, an LDAP client cannot detect it from its operation results.

For the DirX Directory administrator, there are several ways to get information about multiple DSA selection and usage; some of them may require a deeper understanding of networking: The following sections describe these methods.

### 10.6.1. Using the LDAP Exception Logs to Monitor Contact DSAs

When the LDAP server starts, its startup log contains the full list of selectable DSAs as configured in **dirxldap.cfg**.For example:

```
0 "DirX Directory V8.5 64-Bit LDAP Server running.
    OSName=Microsoft Windows 7  64-bit- Service Pack 1 (build 7601)
    HostName=host2, IP4=123.45.67.92, 123.45.67.93,
    Ldap-Port=8080, SSL-Port=636, Rpc-Port=6999, StartTLS=enabled
    Accepted SSL Protocols=SSLv3.0 TLSv1.0 TLSv1.1 TLSv1.2
    Ldap-Cfg=ldapConfiguration, Ldap-Conn-Max:555, Cache=disabled(3),
SSL-CRL-Checking=OFF, PID=1788,
    UID=A412447 (0), EUID=n/a (0), CP:o=my-company
    ThreadPoolSize=32, Audit=on (level:max), SockMode=async,
CtxLimit=12000, IPStack=4, BannedFilterAttr:-none-, FD_SETSIZE=8190"
          -- 0x45046b7f NOTICE   ldap_cfg mn_ldap_listener 1414
0 "LDAP-Server Up and Running...  Using the following DSAs:
    Contact-DSA:Name=/CN=DSA2, enabled=yes, fails=0,
PSAP=TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=4711',DNS='(HOST=ho
st2,SSLPORT=21201,PLAINPORT=21200, MODE=ssl)'
    Contact-DSA:Name=/CN=DSA3, enabled=yes, fails=6,
PSAP=TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=4711',DNS='(HOST=ho
st3,SSLPORT=21201,PLAINPORT=21200, MODE=plain)'"
```

The lines starting with **Contact-DSA** give the names and PSAPs of selectable DSAs. In this example, there are the two DSAs DSA2 and DSA3.

Whenever a new DAP bind fails, the server's exception log indicates to which DSA the bind has failed. For example:

```
99 DRX_Bind(workspace: 0x000000000450AB88   session:
0x000000000450AA48
    res: ???     do_sign: FALSE bound_session: ???) =
DRX_LOCAL_ABORT_RECEIVED(78)
session:         <ABSENT>
res:    ???
bound_session:  ???
     -- 0x4504483e WARNING  api      dx_dap.cpp       1935-3257
30:58:154
99 DRX_Bind failed to selected DSA:
    DSA-Name:"/CN=DSA2"
    DSA-
```

```
Addr:"TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=4711',DNS='(HOST=h
ost2,SSLPORT=21201,PLAINPORT=21200,MODE=plain)'"
```

If you find entries like this one, check the corresponding DSA to make sure it's up, reachable and running properly. If these messages appear for a longer period of time (significantly longer than just for a simple crash plus re-start time), it might be useful to disable this DSA for the lifetime of the LDAP server instance with the extended operation **ldap_disable_config_dsa** and then analyze the DSA problem before re-enabling it. For example, if a DSA from the selection list should be shut down for maintenance, it is a good idea to disable it first before shutting it down to avoid unnecessary bind errors and timeouts that can appear when this DSA gets selected by the round-robin algorithm.

### 10.6.2. Using netstat to Monitor Established Communications

As every DAP bind to a DSA creates a TCP connection first, you can use the **netstat -an** command to check to which DSA the LDAP server is connected and look for the ESTABLISHED connections. For example:

```
TCP    123.45.67.92:21201     123.45.67.92:55925     ESTABLISHED
TCP    123.45.67.92:21201     123.45.67.92:55926     ESTABLISHED
TCP    123.45.67.92:21201     123.45.67.92:55927     ESTABLISHED
TCP    123.45.67.92:21201     123.45.67.92:55928     ESTABLISHED
TCP    123.45.67.92:21201     123.45.67.92:55929     ESTABLISHED
TCP    123.45.67.92:21201     123.45.67.92:55934     ESTABLISHED
TCP    123.45.67.92:53726     123.45.67.93:60001      ESTABLISHED
TCP    123.45.67.92:53915     123.45.67.46:445        ESTABLISHED
TCP    123.45.67.92:53917     123.45.67.99:22        ESTABLISHED
TCP    123.45.67.92:55925     123.45.67.92:21201     ESTABLISHED
TCP    123.45.67.92:55926     123.45.67.92:21201     ESTABLISHED
TCP    123.45.67.92:55927     123.45.67.92:21201     ESTABLISHED
TCP    123.45.67.92:55928     123.45.67.92:21201     ESTABLISHED
TCP    123.45.67.92:55929     123.45.67.92:21201     ESTABLISHED
TCP    123.45.67.92:55934     123.45.67.92:21201     ESTABLISHED
TCP    123.45.67.92:56009     123.45.67.93:443        TIME_WAIT
TCP    123.45.67.92:56010     123.45.67.93:443        TIME_WAIT
TCP    123.45.67.92:56015     123.45.67.16:80         ESTABLISHED
TCP    123.45.67.92:56024     123.45.67.86:445        SYN_SENT
TCP    123.45.67.92:56025     123.45.67.16:80         ESTABLISHED
TCP    123.45.67.92:63697     123.45.67.93:60001      ESTABLISHED
TCP    123.45.67.92:63702     123.45.67.93:60000      ESTABLISHED
TCP    123.45.67.92:63710     123.45.67.86:60001     ESTABLISHED
TCP    123.45.67.92:63756     123.45.67.24:49167      ESTABLISHED
TCP    123.45.67.92:63889     123.45.67.24:49167      ESTABLISHED
```

```
TCP    123.45.67.92:63898     123.45.67.93:60001      ESTABLISHED
TCP    123.45.67.92:64072     123.45.67.15:5061       ESTABLISHED
TCP    127.0.0.1:8307         0.0.0.0:0              LISTENING
TCP    127.0.0.1:9089         0.0.0.0:0              LISTENING
TCP    127.0.0.1:54204        127.0.0.1:54205        ESTABLISHED
TCP    127.0.0.1:54205        127.0.0.1:54204        ESTABLISHED
TCP    192.168.44.1:139       0.0.0.0:0              LISTENING
TCP    192.168.163.1:139      0.0.0.0:0              LISTENING
```

From your configured PSAPs, you can determine by IP and port to which DSAs the LDAP server holds connections and to which it does not.

In the example, we see that only connections to ports 21201 appear but none for 21200, which indicates that something is wrong with /CN=DSA2.

### 10.6.3. Using LDAP Audit

Possibly the best and easiest way to trace the multiple contact DSA feature is to examine the LDAP audit. The header section lists the configured DSAs, which gives you an overview of the available contact DSAs (similar to the LDAP server startup message).

For example:

+

```
Contact-DSA               :Name=/CN=DSA2, enabled=yes, fails=0,
PSAP=TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=4711',DNS='(HOST=ho
st2,SSLPORT=21201,PLAINPORT=21200,MODE=ssl)'
Contact-DSA               :Name=/CN=DSA3, enabled=yes, fails=6,
PSAP=TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=4711',DNS='(HOST=ho
st3,SSLPORT=21201,PLAINPORT=21200,MODE=plain)'
```

If you look at the LDAP audit's bind records, you can discover what happens during DSA selection. For example:

```
----------------- OPERATION 000007 ----------------
  Create Time    :Mon Apr 11 13:54:31.974791 2016
  Start Time     :Mon Apr 11 13:54:31.974831 2016
  Send End Time  :Mon Apr 11 13:54:34.735626 2016
  End Time       :Mon Apr 11 13:54:34.735660 2016
  PoolThread#    :6 (0x1454)
  OpUUID         :ba7966b1-65a6-4b68-a339-8d7cdae464e3
  DapBindId      :000e0006
```

```
  Contact-DSA    :/CN=DSA2
  Concurrency    :1
  OpStackSize    :1
  OpFlow In/Out  :0/0
  Duration       :2.760829 sec
   LDAP QTime    :0.000039 sec
   LDAP Prep Time:2.753598 sec (3 RecvCalls, 0 Wouldblocks)
   LDAP Resp Time:0.000128 sec
    LDAP Snd Time:0.000045 sec (1 SendCalls, 0 Wouldblocks)
    LDAP Enc Time:0.000026 sec
  OP Linger  Time:0.000034 sec
  API Time       :0.007102 sec
   API-Send      :0.000276 sec
   API-ICOM Wait :0.006670 sec
    IDM Time     :0.000123 sec (0 Wouldblocks)
    DSA Time     :0.006500 sec
   API-Recv      :0.000151 sec
    API-Dec      :0.000076 sec
  User           :cn=admin,o=my-company
  IP+Port+Sd     :[127.0.0.1]+56090+844
Op-Name        :LDAP_Con1_Op0
  UniqueOpID     :7
  Operation      :BIND
  Version        :3
  MessageID      :47
  Bind-Type      :simple
  Security       :normal
  DAP-Share-Count:1
  DSA-Retry-Count:1
  DSA-Retry-Dur  :2.760190 sec
  Controls #     :2
    Ctrl Type    :1.3.6.1.4.1.21008.108.63.1 (Session Tracking
Control)
    Critical     :no
    SID-IP       :123.45.67.92
    SID-Name     :DirX Manager 2.3 (Build 82; 2016-01-15 13:20:23)
[4604]
    SID-Oid      :1.3.6.1.4.1.21008.108.63.1.3 (Sasl-Auth-Username)
    SID-Info     :cn=admin,o=my-company
    Ctrl Type    :1.3.6.1.4.1.42.2.27.8.5.1 (Password Policy)
    Critical     :no
```

```
    Ctrl Val(len):0
  Bytes Received :223
  Bytes Returned :64
  Socket Mode    :ssl  Abandoned      :no
  Result Code    :0 (success)
  Error Message  :Bind succeeded.
```

In this example, the field:

```
Contact-DSA :/CN=DSA2
```

indicates the DSA that was (finally) chosen for the DAP backend bind. The fields

```
DSA-Retry-Count:1
DSA-Retry-Dur  :2.760190 sec
```

indicate how many DSAs failed to bind before final succeeding and how much time was spent to iterate through multiple DSAs to get a DAP bind (or all failed).

In this example—where /CN=DSA3 is down—we know that DSA3 was originally selected, that it failed and that the fail took 2.7 seconds before finally DSA2 succeeded. It's easy to see that most of the total time of the bind operation was spent in DAP bind iterations.

Note that if a network is down or a host is not running or a blocking firewall is active and so on, TCP may take some time—typically around 20 seconds—to detect the situation by retransmission and timeout expiration. This action and its subsequent delay cannot be avoided, as this is a TCP procedure and not a server application issue. This means that even if a bind finally succeeds, it might take significant time to succeed from the LDAP client's point of view.

If the first selected DSA succeeds (no further re-tries are necessary), you'll find:

```
DSA-Retry-Count:0
```

For audit records other than bind - for example, search - you can see the DSA in the Contact-DSA field. For example:

```
OPERATION 000008 ----------------
  Create Time    :Mon Apr 11 13:54:34.744390 2016
  Start Time     :Mon Apr 11 13:54:34.744466 2016
  Send End Time  :Mon Apr 11 13:54:34.746446 2016
  End Time       :Mon Apr 11 13:54:34.746456 2016
```

```
PoolThread#    :7 (0xf8)
OpUUID         :d91a1a59-3466-4f9e-889b-01feb16db6d2
DapBindId      :000e0006
Contact-DSA    :/CN=DSA2
Concurrency    :1
OpStackSize    :1
OpFlow In/Out  :0/0
Duration       :0.001990 sec
 LDAP QTime    :0.000076 sec
 LDAP Prep Time:0.000428 sec (3 RecvCalls, 0 Wouldblocks)
 LDAP Resp Time:0.000248 sec
  LDAP Snd Time:0.000058 sec (2 SendCalls, 0 Wouldblocks)
  LDAP Enc Time:0.000046 sec
OP Linger  Time:0.000010 sec
API Time       :0.001312 sec
 API-Send      :0.000065 sec
 API-ICOM Wait :0.001053 sec
  IDM Time     :0.000079 sec (0 Wouldblocks)
  DSA Time     :0.000929 sec
 API-Recv      :0.000188 sec
  API-Dec      :0.000174 sec
User           :cn=admin,o=my-company
IP+Port+Sd     :[127.0.0.1]+56090+844
Op-Name        :LDAP_Con1_Op1
UniqueOpID     :8
Operation      :SEARCH
Version        :3
MessageID      :48
Base Obj       :(ldapRoot)
Scope          :baselevel
FilterLen      :18 (limit:1000)
Filter         :(objectclass=PRES)
Size Limit     :1000
Time Limit     :0
Deref Alias    :never
Types Only     :no
Req Attr #     :11
  Req Attr     :* (all user attributes)
  Req Attr     :namingContexts
  Req Attr     :altServer
  Req Attr     :supportedExtension
```

```
    Req Attr     :supportedControl
    Req Attr     :supportedSASLMechanisms
    Req Attr     :supportedLDAPVersion
    Req Attr     :subschemaSubentry
    Req Attr     :supportedFeatures
    Req Attr     :vendorName
    Req Attr     :vendorVersion
  Found Entries  :1
  Found Attrs    :11
  Found Values   :19
  Op  Ctx Size   :147456 Bytes
  API Ctx Size   :81920 Bytes
  All Ctx Size   :55 MB
  Controls #     :1
    Ctrl Type    :1.3.6.1.4.1.21008.108.63.1 (Session Tracking
Control)
    Critical     :no
    SID-IP       :123.45.67.92
    SID-Name     :DirX Manager 2.3 (Build 82; 2016-01-15 13:20:23)
[4604]
    SID-Oid      :1.3.6.1.4.1.21008.108.63.1.3 (Sasl-Auth-Username)
    SID-Info     :cn=admin,o=my-company
  Bytes Received :382
  Bytes Returned :693
  Socket Mode    :plain
  Cached Result  :no
  Abandoned      :no
  Result Code    :0 (success)
  Error Message  :Search succeeded. Found 1 Entries (0 Aliases), 11
Attributes, 19 Values. (ChainedResult=no)
```

### 10.6.4. Using DirX Directory Extended Operations

You can use the **dirxextop ldap_show_config_dsas** extended operation to monitor which DSAs are configured, which DSA is to be selected next, which DSAs are currently enabled and how many failures occurred when DAP binds were performed. The description of the **dirxextop** command in the *DirX Directory Administration Reference* provides command syntax and usage for the operation.

Here is example output returned by the operation:

```
List of configured Contact-DSAs for LDAP server on 'host2' at Tue May
```

```
17 14:21:26.335769
=========================================================================
==========
 DSA-Name:/CN=DSA2
    Status      :enabled
    PSAP
:TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=4711',DNS='(HOST=host1,
SSLPORT=21201,PLAINPORT=21200,MODE=ssl)'
    BindFails   :2
    PermDisables:0
    TempDisables:2  (Last: Tue May 17 14:21:15.753710)
    ReEnables   :1
    Selections  :5
-------------------------------------------------------------------------
----------

(*) == DSA to be selected for next Backend-Bind
```

## 10.7. Mixing Single and Multi-Contact DSA Configurations

Recall from the section “Setting up Multiple LDAP Servers” in the chapter “Extending the DirX Directory Service” that multiple LDAP servers can be set up on a single machine to handle the requirements of particular LDAP clients.These additional LDAP servers can be configured to read their contact DSA information from their own LDAP server configuration files, allowing each additional LDAP server to use either a multiple contact DSA configuration or a single contact DSA configuration depending on its requirements.

When an LDAP client in a multiple contact DSA configuration needs to receive its data from a dedicated DSA and the primary LDAP server in this configuration is set up to use multiple contact DSAs, an additional LDAP server needs to be configured that uses the dedicated DSA as its single contact DSA and reads its contact information about this DSA from a separate LDAP configuration files that is specific to this additional LDAP server.

For example, suppose the administrator of DSA2 has set up the Nagios monitoring environment with the intention of evaluating DSA2’s performance over time.The primary LDAP server – the first server to be set up – uses the multiple contact DSA configuration because it’s fielding calls from LDAP clients for data-centric operations on DSA2’s DIT.To support the Nagios environment, the DSA2 administrator now needs to set up an additional LDAP server to handle Nagios communication to and from DSA2. This LDAP server’s configuration needs to specify DSA2 as the only contact DSA to ensure that the performance information returned by calls from Nagios plugins is always coming from this particular DSA.

To set up the additional Nagios-specific LDAP server, the DSA2 administrator must perform

the following tasks:

- Set up the additional LDAP server to be dedicated to handling Nagios communications
- Set up a specific LDAP configuration file for this dedicated server
- Set the DirX Directory environment variable that permits LDAP servers to read their own LDAP configuration files
- Re-start the DirX Directory service

### 10.7.1. Creating the Additional LDAP Server

To create the additional LDAP server, the DSA2 administrator follows the procedure described in the section “Setting up Multiple LDAP Servers” in the chapter “Extending the DirX Directory Service” in this guide. The common name that the DSA2 administrator uses for this additional LDAP server’s configuration subentry is **dsa2ldapConfig2**.

### 10.7.2. Creating the Additional Server’s LDAP Configuration File

Next, the DSA2 administrator creates an individual LDAP configuration file for the additional LDAP server that specifies DSA2 as the only contact DSA. The file name is in the format:

**dirxldap.cfg**[.*subentry_name*]

where *subentry_name* is the common name of the LDAP server’s configuration subentry that corresponds to the new LDAP server. In this scenario, it is **dsa2ldapConfig2**:

```
dirxldap.cfg.dsa2ldapConfig2
```

### 10.7.3. Setting the Environment Variable

To enable the new LDAP server to read its own LDAP configuration file instead of the common **dirxldap.cfg** file, the DSA2 administrator must set the DirX Directory environment variable **DIRX_LDAP_USE_SEPARATE_CLCFG_FILE** before re-starting the DirX Directory service:

```
DIRX_LDAP_USE_SEPARATE_CLCFG_FILE=1
```

The additional Nagios-specific LDAP server will read the configuration file **dsa2ldapConfig2** when it starts up again.

### 10.7.4. Re-starting the DirX Directory Service

Now the DSA2 administrator can re-start the DirX Directory service as described in the section “Setting up Multiple LDAP Servers” in the chapter “Extending the DirX Directory Service”.

# 11. Using LDIF Files for Data Synchronization

As noted in the chapter “Introducing DirX Directory” in the *DirX Directory Introduction*, the DirX Directory DSA can export all or part of its DIT into LDIF content or change file format. DirX Directory administrators can use these LDIF files to replicate changes to a DirX Directory DSA’s directory database with DirX Identity to non-DirX Directory directories such as Windows, Active Directory, NDS, Oracle and Lotus Notes; this process is called directory synchronization and is described in detail in the *DirX Identity* documentation set. DirX Directory administrators can also use LDIF file generation as part of an overall backup strategy. For example, they can import a previously generated LDIF file with **dirxmodify** to restore parts of a DirX Directory DSA’s directory database to a previous state, should they become inconsistent.

This chapter describes how to set up a DirX Directory DSA to generate LDIF content and change files and how to import them into a DirX Directory DSA. It describes a sample LDIF file configuration in which the sample stand-alone DSA established in the chapter “Setting up the DirX Directory Service” is the LDIF file supplier. This chapter provides a brief description of LDIF content and change files. It continues to use the fictitious company “My-Company” to illustrate the planning decisions and administrative tasks necessary to set up the sample LDIF file configuration.

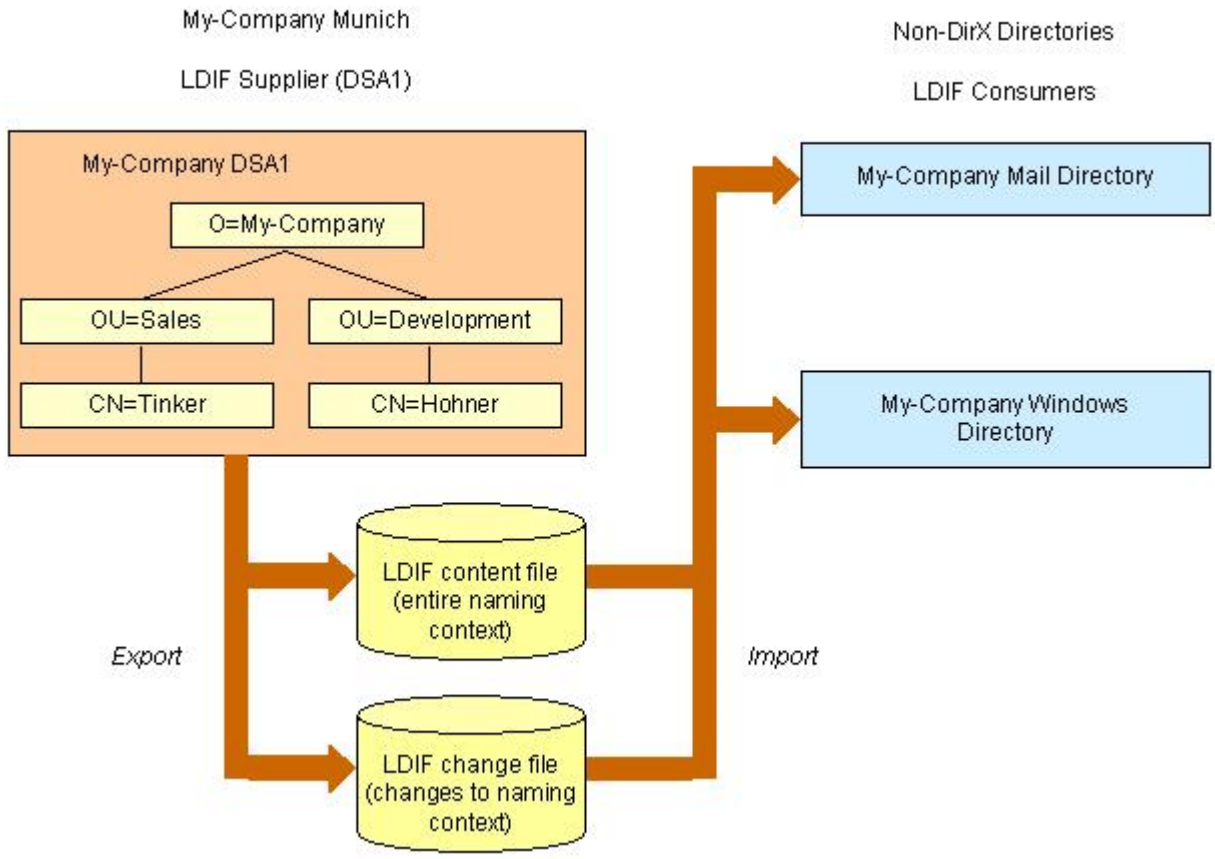
Tcl scripts for building this sample LDIF file configuration and adding the extensions to it can be found in *install_path*/**scripts/LdifAgreements**. These scripts contain all the commands and procedures discussed in this chapter.

## 11.1. Exporting Directory Data to LDIF Files

My-Company has established a single stand-alone DSA that contains general information about its employees.The DSA is the central repository for this employee data and is the site at which the latest changes are recorded.Each employee has a Windows account and a mailbox in the company’s mail system.General information about the employees is stored in Windows and the mail directory.The site administrators want to ensure that the employee data that is duplicated in the Windows and mail directories is periodically updated with the latest changes to the user information in the DSA employee database.It is not necessary that the information in the non-DirX Directory directories is always current with the DSA; the non-DirX Directory directories can be out of synchronization with the DSA for a day or so.

One solution that meets these criteria is to extract the information contained in the DSA that is relevant for directory synchronization into local LDIF content and change files.The administrators of the Windows and mail programs can then pick up and import these files into the non-DirX Directory directories at their convenience, using tools such as DirX Identity that can handle the import of LDIF-formatted files.The following figure illustrates this solution.

*My-Company LDIF Configuration*




### 11.1.1. Building the LDIF Configuration

To build the LDIF configuration, the My-Company DSA administrator and the directory synchronization administrator must first define a directory synchronization process and determine what entries and entry attributes are to be written to the LDIF files. Next, the My-Company DSA administrator must create an **LDIF agreement** that meets the needs of the synchronization process.

An LDIF agreement is a type of Shadow Operational Binding (SOB) that is analogous to a shadowing agreement. A shadowing agreement establishes a shadow supplier and a shadow consumer. In an LDIF agreement, the DSA is the LDIF file supplier and the directory synchronization tool is the LDIF file consumer. An LDIF agreement has some restrictions:

- In an LDIF agreement, there is only a supplier DSA that writes the information to be replicated into local LDIF-formatted files; there is no consumer DSA. Instead, the generated LDIF files are consumed by the directory synchronization tool.

So, for an LDIF agreement:

- The supplier DSA is the DSA that is to write the information into local LDIF-formatted files; this process is always supplier-initiated
- The unit of replication specifies what is to be written to the LDIF files
- The update strategy can be "on change" or "scheduled". A scheduled update strategy must specify at least the agreed-upon interval at which updates are to be written into

LDIF files.

The administrators must first negotiate the unit of replication, the update strategy, and the LDIF agreement ID. They can also decide to establish LDIF agreement policies. The administrator of DSA1 then uses the **dirxadm lob** object and operations locally to create and establish the LDIF agreement. When he completes this procedure, the following process occurs:

- DSA1 writes an **LDIF content file** that contains a list of directory entries in the unit of replication and their attributes. Each entry consists of a distinguished name and a list of attributes, where each attribute has a prefix and one or more values. For My-Company, the LDIF content file consists of all the user information held by DSA1, but modified with an LDIF policy established in the LDIF agreement.
- After the LDIF content file is created, the LDIF agreement remains active. At the agreed-upon scheduled update interval, the DSA writes any changes made to data within the unit of replication on DSA1 to an LDIF change file. Each entry in the LDIF change file contains a special "changetype" attribute that indicates the type of directory modification to be replicated in the non-DirX directory:
  - add a directory entry
  - delete a directory entry
  - modify one or more attributes of a directory entry
  - modify the distinguished name of a directory entry

  The DSA continues to create new LDIF change files at the agreed-upon scheduled interval for as long as the agreement remains active. If the update strategy is "scheduled" and no changes to the unit of replication occur between update intervals, the DSA creates a change file that does not contain any directory information.
- If the LDIF agreement is set to inactive on DSA1, the DSA stops creating LDIF change files.

To establish the LDIF agreement on DSA1, the DSA1 administrator must perform the following tasks:

1. Negotiate the LDIF agreement with the directory synchronization administrator.
2. Create the LDIF agreement on DSA1.
3. Activate the LDIF agreement on DSA1.

The next sections describe how to perform these tasks.

#### 11.1.1.1. Negotiating the LDIF Agreement

First, the DSA1 administrator and the synchronization administrator must agree upon the details of the LDIF agreement over the phone, through email, or by meeting in person. These details include:

- Determining the unit of replication. This decision affects the values supplied in the shadowing-subject (SS) attribute of the LDIF agreement; see the section "Creating the

LDIF Agreement on DSA1" for further details.

- Determining the update strategy. This decision affects the values supplied in the shadowing-subject's update-mode (UM) component; see the section "Creating the LDIF Agreement on DSA1" for further details.
- Determining the LDIF policies. This decision affects the values supplied in the sob-ldif-supplier-policies (LDSUPP) attribute of the LDIF agreement; see the section "Creating the LDIF Agreement on DSA1" for further details.
- Determining the DSA's partial entry shadowing policy.
- Determining the LDIF agreement ID.

##### 11.1.1.1.1. Determining the Unit of Replication

For My-Company, the unit of replication is the entire naming context **/O=My-Company**. The administrators plan to establish an LDIF policy that restricts what the DSA extracts from the naming context.

##### 11.1.1.1.2. Determining the Update Strategy

The update strategy can be "on change" basis or "scheduled". A scheduled incremental update strategy can include a start time for generating the first total LDIF content file, a window during which the DSA will generate LDIF change files, and an update interval at which incremental updates are written to LDIF change files. The interval at which updates are to be created is the sum of the window (WS) and update interval (UI).

The DSA1 administrator and the directory synchronization administrator decide to define a scheduled update strategy in which the DSA creates the LDIF content file immediately upon the agreement's activation (instead of selecting a specific start time for its creation) and creates subsequent LDIF change files every 24 hours.

##### 11.1.1.1.3. Determining the LDIF Policy

Administrators can apply policies to an LDIF agreement that control various aspects of the LDIF file generation procedure. An LDIF policy controls three main areas of the LDIF file generation procedure:

- The information to be extracted from the naming context
- The format of the generated LDIF files
- The procedure used to export the generated LDIF files

Policies for entry extraction can include:

- Whether entries are extracted with only their user attributes and the operational attributes createTimestamp (CRT) and modifyTimestamp (MT), or whether all entry information is extracted, including all operational attributes, references, and subentries (controlled with the ENTRYO component of the LDSUPP attribute)
- Whether modified and deleted entries and their attribute values are extracted (controlled with the MODAT and DELAT components of the LDSUPP attribute)
- Whether the DSA creates an LDIF content file when the LDIF agreement is activated, or

whether it creates only change files (controlled with the CHANGEO component of the LDSUPP attribute)

Policies for generated LDIF file format can include:

- How attribute values are to be encoded (controlled with the BASE64 and CS components of the LDSUPP attribute). The choice of attribute value encoding depends upon how LDIF files are to be processed. The default codeset is UTF-8. However, some synchronization tools require attributes to be encoded in Latin-1, so the administrator can use the codeset (CS) component to specify Latin-1 encoding.
- The characters used to separate entries in the file (the CRLF component of LDSUPP) and the maximum number of characters per line (the MAXROW component of LDSUPP). Administrators can use these components to control LDIF file readability on different operating systems. For example, specify only a carriage return to make files readable on Linux systems, specify carriage return and linefeed to make LDIF files readable on Windows systems.

Policies for controlling the export of LDIF files (controlled with the PROG component of the LDSUPP attribute) can include:

- The name of an executable program to be invoked after an LDIF file is generated and to which the newly created LDIF file is to be redirected
- The name of an executable program to be invoked before the LDIF agreement is activated that performs pre-processing functions such as LDIF file cleanup or switching from LDIF change file format to content format, and then activates the LDIF agreement

The DSA1 administrator and the directory synchronization administrator decide to define an LDIF policy that extracts only entries, their user attributes and their operational attributes createTimestamp (CRT) and modifyTimestamp (MT) into the LDIF content and change files.

#### 11.1.1.1.4. Determining the DSA's partial entry shadowing policy

If your DSA writes LDIF content files and contain huge entries that cannot be loaded as a whole in memory, for example entries with groups that contain more than 1000 members, you must specify the value **TRUE** for the **Partial Entry Shadowing Policy** attribute. (See section "Partial Entry Shadowing Policy" in the *DirX Directory Syntaxes and Attributes* for details.)

#### 11.1.1.1.5. Determining the Agreement ID

An LDIF agreement requires the presence of an agreement ID that must be known by the LDIF file supplier DSA and the administrators who want to use the generated LDIF files. Administrators who want to import the LDIF files can use the agreement ID to determine which LDIF files belong to their LDIF agreement. It is up to the administrators to negotiate the value of an agreement ID before an LDIF agreement can be administered on supplier DSA.

An agreement ID is a sequence of two integers (an identifier value and a version number) separated by a comma. The version number is managed by the DSA. The DSA assigns the

value **0** when the LDIF agreement is created (with **dirxadm lob create**); each time the agreement is established (with **dirxadm lob establish**), the DSA increments the version number by 1. The administrators choose the agreement ID **88** to represent this LDIF agreement.

#### 11.1.1.2. Creating the LDIF Agreement on DSA1

The next step in building the LDIF configuration is for the My-Company administrator to bind to **dirxadm** and create the LDIF agreement on DSA1. To create the LDIF agreement on DSA1 (the local, or "bound to" DSA) with **dirxadm**, the administrator uses the command:

**lob create**
**-agreementid** *agreement_id*
**-status** *coop_status*
**-agreement** *agreement*
**-pol** *sob_policies*

For the agreement on DSA1, the administrator supplies the following values:

- The agreement identifier **88** to the **-agreementid** option
- The keyword **NONCOOPERATIVE** to the **-status** option, which postpones creation of the LDIF content file until the My-Company administrator explicitly establishes the LDIF agreement with the **lob establish** operation
- The shadowing-subject (SS) operational attribute to the **-agreement** option, which should define at least the following:
  - The name of the context prefix (CP) to be replicated in the AREA component. For this LDIF agreement, it is My-Company DSA1's context prefix, which is **/O=My-Company**.
  - The entries belonging to this naming context to be replicated in the replication-area (RA) subcomponent. For this LDIF agreement, the value should be DEF=TRUE, which means that the whole naming context must be replicated.
  - The attributes to be replicated in the ATT subcomponent. For this LDIF agreement, the value should be DEF=TRUE to indicate that all attributes are to be replicated.
  - A periodic-strategy (PS) in the update-mode (UM) that schedules the creation of LDIF change files every 24 hours (specified as 86400 seconds in the window-size (WS) subcomponent)
- The sob-ldif-supplier-policies (LDSUPP) operational attribute to the **–pol** option, which specifies that entries, their user attributes, and their operational attributes createTimestamp (CRT) and modifyTimestamp (MT) are to be written to the LDIF files (ENTRYO=TRUE). Subentries, references, and other operational attributes are not to be written.

The following **dirxadm** command creates the LDIF agreement on DSA1:

```
lob create -agreementid 88  \
           -status noncooperative         \
```

```
               -agreement {SS={AREA={CP={/O=My-Company},
                                      RA={DEF=TRUE}
                                     },
                                 ATT={DEF=TRUE}},
                              UM={SI={S={PS={WS=86400,
                                   UI=0}}}}
               -pol {LDSUPP={ENTRYO=TRUE}} \
```

If this command completes successfully, it displays the following information:

```
AGR={ID=88,VERS=0}
```

#### 11.1.1.3. Activating the LDIF Agreement on DSA1

The final step in building the LDIF configuration is to bind to the DSA with **dirxadm** and activate the LDIF agreement so that DSA1 starts to generate the LDIF files.

To activate an LDIF agreement with **dirxadm** on a local ("bound to") DSA, use the command:

**lob establish -agreementid** *agreement_id*

This command sets the LDIF agreement to the status **COOPERATIVE**. The following **dirxadm** command activates the My-Company LDIF agreement on DSA1:

```
   lob establish -agreementid 88    \
```

Because the scheduled update strategy for the My-Company LDIF agreement does not specify a start time at which the LDIF content file is to be generated, DSA1 creates an LDIF content file immediately. It creates LDIF change files every 24 hours for as long as the LDIF agreement remains enabled.

### 11.1.2. Exporting LDIF Content and Change Files

When an LDIF agreement is activated, the DSA writes an LDIF content file and subsequent LDIF change files to the directory *dirx_install_path***/server/ldif**. It names LDIF files in the format:

*agreement_id.timestamp*

where *agreement_id* is the LDIF agreement ID without the version number that was specified in the **-agreementid** option to the **lob establish** operation (or the **lob create** operation, if the **-status** option was set to **COOPERATIVE**) and *timestamp* is the time at which the LDIF file was created. Both *agreement_id* and *timestamp* are supplied with leading zeros to establish a 14-digit string. For example, the My-Company LDIF agreement

files are named:

**00000000000088.00000936197500**

When a DSA is in the process of creating an LDIF file, but has not completed it, the character **p** is appended to the filename. The filename then appears in the format:

*agreement_id.timestamp***p**

For example, while creating the My-Company LDIF file above the file is named:

**00000000000088.00000936197500p**

Unless a policy for exporting LDIF files has been specified in the LDIF agreement, administrators who want to obtain the LDIF files for import into their directories must explicitly check for newly created LDIF files and copy them to the target system or directory, or create an application that performs this task. In addition, the DSA does not delete LDIF files unless an LDIF policy for controlling the export of LDIF files (PROG) has been specified in the LDIF agreement. Consequently, it is the responsibility of the administrators to delete LDIF files once they are no longer useful, or create an application that deletes them.

In the case of My-Company, no LDIF policy for exporting LDIF files has been defined in the LDIF agreement. As a result, the My-Company DSA1 administrator and the administrators of the non-DirX Directory directories must ensure that the LDIF files are copied to the appropriate locations and that LDIF content and change files are removed from the DSA.

## 11.1.3. Managing LDIF Agreements

The **dirxadm** administration tool allows you to maintain LDIF agreements once they have been created and to remove them entirely from a DSA. The next sections describe how to display, modify, disable, enable, deactivate, and remove LDIF agreements with **dirxadm**. All sections assume that the administrator has already bound to the DSA.

### 11.1.3.1. Displaying LDIF Agreements

You can use **dirxadm** to view the properties of existing LDIF agreements.

To display an LDIF agreement with **dirxadm** on the local ("bound to") DSA, use the command:

**lob show -agreementid** *agreement_id*

For example, the My-Company administrator can issue the command:

```
lob show -agreementid 88 \
        -pretty
```

to display the contents of the LDIF agreement ID 88 on My-Company DSA1:

```
    Validity-Agreement
    Operational-Binding-State     : COOPERATIVE
    OprBindMngmnt-Validity
      Validity-From               : 990901152622Z
      Agreement
        Shadow-Subject
          Area-Specification
            Context-Prefix        : /O=My-Company
            Replication-Area
              Default-Value       : TRUE
          Attribute-Selection
            Default-Value         : TRUE
          Knowledge
        Update-Mode
          Supplier-Initiated
            Scheduled
              Periodic-Strategy
                Window-Size       : 86400
                Update-Interval   : 0
        Secondary-Shadows         : FALSE
    Update-Status
      Disabled                    : FALSE
      Update-Status
        Supplier-Update-Status
          Area-Change-State       : 1
          Old-Updates
            Segment-ID            : 0
            Update-Time           : 19990901152622Z
            Area-Changes          : TRUE
    Related-Policies
      LDIF-Supplier
        Entry-Only                : TRUE
```

#### 11.1.3.2. Enabling and Disabling an LDIF Agreement

You can use **dirxadm** to disable and enable LDIF agreements. Disabling an LDIF agreement stops the DSA from creating any further LDIF change files, but does not change the agreement's status (changes to the directory continue to be recorded). When the LDIF agreement is re-enabled, the DSA begins to generate LDIF change files again. You can only disable and enable activated LDIF agreements (agreements whose status is set to **COOPERATIVE**).

To disable an LDIF agreement with **dirxadm**, use the command:

**lob disable –agreementid** *agreement_id* [**-supplier** *dsa_name*]

For example, the DSA1 administrator can issue the following **dirxadm** command to disable the LDIF agreement on DSA1 (the "bound to" DSA):

```
lob disable -agreementid 88
```

To enable a disabled LDIF agreement with **dirxadm**, use the command:

**lob enable -agreementid** *agreement_id* [**-supplier** *dsa_name*]

For example, the DSA1 administrator can issue the following **dirxadm** command to enable the LDIF agreement on DSA1 that he has previously disabled:

```
lob enable -agreementid 88
```

#### 11.1.3.3. Deactivating an LDIF Agreement

You can use **dirxadm** to deactivate an LDIF agreement. Deactivating an LDIF agreement changes the LDIF agreement's status from **COOPERATIVE** to **NONCOOPERATIVE** and stops the DSA from creating any further LDIF change files.

Note that you must reactivate (establish) the agreement to change its status from **NONCOOPERATIVE** to **COOPERATIVE**. Each time a deactivated (terminated) LDIF agreement is reactivated (established), the DSA generates a new LDIF content file, unless the update strategy is "changes only" (CHANGEO=TRUE). With this strategy, a content file is not created.

To deactivate an LDIF agreement with **dirxadm**, use the command:

**lob terminate -agreementid** *agreement_id* [**-supplier** *dsa_name*]

The following **dirxadm** command deactivates the LDIF agreement on DSA1 (the "bound to" DSA):

```
lob terminate -agreementid 88
```

#### 11.1.3.4. Removing an LDIF Agreement from a DSA

You can use **dirxadm** to remove the contents of an LDIF agreement completely from the DSA. Before you can remove an LDIF agreement, you must first deactivate it with **dirxadm**.

To remove an LDIF agreement from a DSA with **dirxadm**, use the command:

**lob delete –agreementid** *agreement_id* [**-supplier** *dsa_name*]

The following **dirxadm** command removes the My-Company LDIF agreement from DSA1 (the "bound to" DSA):

```
lob delete -agreementid 88
```

## 11.2. Importing Directory Updates from an LDIF File

You can use **dirxmodify** to update a populated DirX directory from an LDIF content or change file.For example, suppose My-Company acquires a small network switch manufacturing company in Stuttgart.My-Company plans to retain the employees of this company and wants to centralize the Stuttgart employee information in the stand-alone DSA in Munich.The Stuttgart company uses a non-DirX Directory LDAP directory; the DSA1 administrator in Munich asks the Stuttgart company's directory administrator to create an LDIF content file of the company's LDAP directory data and send it to him.

The structure of the LDAP directory's schema is essentially the same as My-Company's DSA schema, with the exception of the attribute type names for telephone and mobile phone.These attribute type names will need to be changed to the names used in My-Company's DSA schema.The My-Company administrator also decides that he does not want to import the information given in the LDAP directory's description and business category attributes.

To load the LDIF content file to DSA1 with **dirxmodify**, the administrator uses the command:

**dirxmodify -f** *LDIF_file*
**-D** *bind_id*
**-w** *password*
**-s** *separator-old_type-separator-new_type*
**-i** *attribute_type* [...]
**-e** *error_file*

Where:

- **-f** specifies the pathname or relative name of the content or change file to be imported
- **-D** specifies the distinguished name part of the authentication credentials that **dirxmodify** is to use to bind to the DirX Directory LDAP server, in LDAP format
- **-w** specifies the password part of the **dirxmodify** bind authentication credentials
- **-s** replaces all occurrences of an attribute type name with a new type; a separator like a slash (/) or a colon (:) is used to separate the old name from the new one (for example, /handy/mobile).
- **-i** specifies an attribute type that **dirxmodify** is to ignore when importing the contents of the LDIF file.
- **-e** specifies the name of a file in which **dirxmodify** will record any LDIF file entries that it cannot process.

For the import to DSA1, the administrator supplies the following values:

- The file name **Stuttgart_Mfg.ldif** to the **-f** option
- The distinguished name **cn=admin,o=My-Company** to the **-D** option
- The password **dirx** to the **-w** option
- The attribute type **telephone** to replace **phone** and the attribute type **mobile** to replace **handy** to the **-s** option
- The attribute types **description** and **businessCategory** to the **-i** option

The following **dirxmodify** command loads the **Stuttgart_Mfg.ldif** file into DSA1:

```
dirxmodify -f Stuttgart_Mfg.ldif
           -D cn=admin,o=My-Company
           -w dirx
           -s /phone/telephone
           -s /mobile/handy
           -i description
           -i businessCategory
           -e Stuttgart_Mfg.err
```

# 12. Monitoring DirX Directory

This chapter describes how to use the monitoring tools provided by DirX Directory, including information about:

- How to monitor LDAP and DAS MIBs.
- How to use LDAP extended operations for monitoring issues.
- How to use SNMPv2 traps for monitoring issues.
- How to use the DirX Directory Nagios plugins to monitor DirX Directory processes and service and system resources within a Nagios® Core™ monitoring system.
- How to use the DirX Directory Supervisor to monitor operations in floating master configurations.
- How to use LDAP and DSA auditing.
- How to use diagnostic logging.
- How to monitor DirX directory database consistency and operating issues.

## 12.1. Monitoring the DSA MIBs with dirxadm

When a DSA is started, the DirX Directory service begins to write statistical information about the DSA and other OSI applications into two in-memory MIBs:

- The Network Services, or Application MIB
- The Directory, or DSA MIB

DirX Directory administrators can monitor the performance of DirX Directory by periodically examining the data in these MIBs.The tools for examining the DirX Directory MIBs include:

- The **dirxadm nmi** object and its operations
- The LDAP extended operations for DirX Directory MIB display available through the **dirxextop** command (see the **dirxextop** section in the *DirX Directory Administration Reference* for details)
- DirX Directory Manager's Monitor view (see the *DirX Directory Manager Guide* for details)

The next sections describe how to use **dirxadm nmi** commands to display the Application and DSA MIBs. The sections also provide some hints on how to use the data within the MIBs to evaluate DirX Directory performance. For a description of all the **dirxadm nmi** operations, see the **dirxadm** section in the *DirX Directory Administration Reference*.

### 12.1.1. Examining the Application MIB

The Application MIB is a common storage area to which statistics about all OSI applications are written. The Application MIB consists of two tables:

- The association table, which stores statistics about the associations (connections) made

by OSI applications. Information in the association table includes a count of the number of associations made and information about the current association, such as the protocol in use, the name and type of the application that opened the association and the duration of the association.

- The application table, which stores information about the OSI applications making the associations, and stores statistics about their associations with other applications. Information in the application table includes the names and version numbers of the OSI applications, their accumulated inbound and outbound associations, and the current number of associations they have open. Although there are potentially many OSI applications in the network for which this table should contain entries, the DirX Directory DSA holds information about only one OSI application - itself. Consequently, when you view the application table with **dirxadm**, you see just one table entry, which is that of the DirX Directory DSA.

The Application MIB provides details about the DSA installation, such as the name and version number of the DSA installation (for example, **DirX** and **1.0**), the application entity title, (which is the distinguished name of the DSA), and the time the DSA was last restarted. It also provides information that you can use to determine how intensively the DSA has been used and to make decisions about the installation based on this information. For example, if the Application MIB shows a large number of inbound associations, you may want to recommend to management that they charge usage fees for DSA operations and request larger machines for improved support of the DSA installation.

To display the entire Application MIB, use the **dirxadm** command:

**nmi show -mib APP -pretty**

To display just the application table portion of the Application MIB, use the **dirxadm** command:

**nmi show -mib APP -table APPLICATION -pretty**

To display just the association table portion of the Application MIB, use the **dirxadm** command:

**nmi show -mib APP -table ASSOCIATION -pretty**

Using the **-pretty** option with these commands generates more easily readable output. You do not need to use the **-pretty** option if you are planning to redirect the output to another program, for example, a table-formatting program.

### 12.1.2. Examining the DSA MIB

The DSA MIB provides statistics about the DSA since it was last started. The DSA MIB consists of three tables:

- The entries table, which provides information about the entries in the DSA's DIT
- The interaction table, which provides information about the interactions between this DSA and other DSAs
- The operations table, which provides information about the operations initiated by

users bound to the DSA, by other DSAs bound to the DSA, and by the DSA itself.

To display the entire DSA MIB, use the **dirxadm** command:

**nmi show -mib DSA -pretty**

Using the **-pretty** option with these commands generates more easily readable output. You do not need to use the **-pretty** option if you are planning to redirect the output to another program, for example, a table-formatting program.

#### 12.1.2.1. Examining the Entries Table

The entries table provides statistics about the entries in the portion of the DIT owned by the DSA. This information includes:

- A count of the total number of entries mastered by the DSA
- A count of the total number of entries shadowed by the DSA
- Caching statistics; for example, a count of read operations on shadow entries (slave hits)

To display just the entries table, use the **dirxadm** command:

**nmi show -mib DSA -table ENTRIES -pretty**

You can use the information in the entries table to keep track of the size of the DIT managed by the DSA.

#### 12.1.2.2. Examining the Interaction Table

The interaction table provides information about outgoing interactions with other DSAs (one set of information per DSA). You can use this table to monitor the success of DSP and DISP interactions. If the table indicates failures, you can consult the exception log file (named **USR***processid*) to determine the cause of the failure.

To display just the interaction table, use the **dirxadm** command

**nmi show -mib DSA -table INTERACTION -pretty**

#### 12.1.2.3. Examining the Operations Table

The statistics in the operations table include:

- Counts of the anonymous and authenticated binds performed to the DSA and a count of the bind security errors that have occurred
- Counts of the DAP operations, such as read, compare, search, and so on, performed on the DSA
- Counts of the referrals and chainings returned
- Counts of security errors and DSA errors that have occurred

To display just the operations table, use the **dirxadm** command:

**nmi show -mib DSA -table OPERATIONS -pretty**

The statistics about DAP operations in the operations table can provide you with a profile of how users are using the DSA.For example, you can compare how many occurrences of one type of operation there are compared to occurrences of other types, for example, modifies to searches.

The statistics about chaining and referrals in the operations table can help you to evaluate your DSA configuration.For example, a high number of chains or referrals returned indicates that you may need to shadow some other DSA.

Finally, the error statistics in the operations table can help you to evaluate the DSA's security.For example, a high number of rejected inbound associations in the operations table might indicate that unauthorized and potentially untrustworthy users are trying to connect to your DSA.A high bind security error count in the operations table could mean that users are trying to guess passwords.

### 12.1.3. Initializing the DirX Directory MIBs

The DirX Directory service continues to write information to the Application and DSA MIBs until the DSA is stopped.Consequently, you initialize the DirX Directory MIBs by stopping the DSA with the **dirxadm stop** command, and then restarting the DSA with the **dirxadm start** command.The **sys (dirxadm)** reference page in the *DirX Directory Administration Reference* provides further information about the **dirxadm start** and **stop** commands.

## 12.2. Monitoring the LDAP MIBs

When the LDAP server starts, it automatically creates several MIB tables that contain information about static and dynamic processing data.The tables are automatically maintained and need no further administration.These tables are:

- The **static** table - Use this table to examine quickly most of the configuration settings of a running server, such as port numbers, server type, supported controls and so on.
- The **total** table - Use this table to get information about the number of processed operations in total or detailed by means of operation types.For example, this table counts how many operations are processed, how many binds, searches, adds, and so on.
- The **current** table - Use this table to get information about the current status of how many LDAP/DAP connections are established, the current cache status, the thread pool activity and about the established socket connection states, including:
- The **assoc** table - provides a detailed listing of important LDAP connection parameters and provides information about operations that are currently running (for example, search parameters).
- The **env** table - contains all the environment variable settings that are currently visible to the LDAP server

You can access all of these tables using the LDAP extended operations ldap_mib_static, ldap_mib_total, ldap_mib_current, ldap_mib_assoc, ldap_mib_env and ldap_mib_dump.(See the **dirxextop** reference page in the *DirX Directory Administration*

*Reference* for details.)

## 12.3. Monitoring DirX Directory with LDAP Extended Operations

DirX Directory administrators can also use the **dirxextop** command to run LDAP extended operations that display monitoring and diagnostic information about DirX Directory processes as well as perform some enable/disable functions.You can find a description of LDAP extended operations in Section 4.12 "Extended Operations" in the document Lightweight Directory Access Protocol (v3) RFC 2251, December 1997. The extended operations provided by DirX Directory are intended for support purposes and are described in the **dirxextop** reference page in the *DirX Directory Administration Reference*.

Some of these functions can also be performed via **dirxadm**, a DirX Directory command, or a system command; for example, **dirxadm ldap audit info**.However, rather than using internal RPC protocols and ports, as with **dirxadm**, LDAP extended operations are performed as LDAP client operations sent over LDAP(v3).If you are running a DirX Directory installation in a firewall configuration, **dirxextop** can be a better monitoring and diagnostic option because it uses a well-known LDAP port that is usually open in a firewall.

The **dirxextop** command performs a single LDAP extended operation.See the **dirxextop** reference page in the *DirX Directory Administration Reference* for details.

## 12.4. Monitoring DirX Directory with SNMPv2 Traps

SNMPv2 traps provide another mechanism for monitoring the DirX Directory service.An SNMP trap (also called an alarm) is an unsolicited message sent over UDP/IP to a SNMPv2 receiver application to notify it of specific situations.(See RFC 1155 and RFC 1905 for details.) DirX Directory can be configured to send out SNMPv2 traps on conditions that require administrator attention, such as security-related events or critical operational status.

Because UDP is a connectionless protocol, SNMP traps are lost if there is no receiver running and listening for them.As with any other UDP protocol, SNMP traps can be lost within a network under certain circumstances, such as dropouts or network congestion due to network overload or similar events.Therefore, the sender (DirX Directory, in this case) cannot determine whether the trap has successfully reached its receiver.This is a limitation of the underlying transport UDP, not of DirX Directory.

DirX Directory administrators can use DirX Directory SNMPv2 traps to monitor and detect problems in DirX Directory service operations.The traps that DirX Directory can send include:

- Traps that indicate normal service processing, for example, starting and stopping server processes.
- Traps that indicate temporary problems, for example lack of resources.
- Traps that indicate permanent problems; for example, conditions that result in a server exit without restarting the server process.
- Traps that can help to control DirX Directory service performance; for example, by

indicating when a directory operation exceeds a specific time limit.

You (the DirX Directory administrator) are responsible for providing the application that receives the DirX Directory SNMPv2 traps. For example, on Linux systems, you can use the **trapd** daemon process of the **netsnmp** package. You can also use the **DIRX-MIB.txt** file (provided with DirX Directory) in your trap receiver application to provide more readable output of DirX Directory SNMPv2 traps; for example, trap names instead of OIDs.

To enable SNMPv2 traps:

- Edit the SNMPv2 trap configuration file *install_path***/conf/snmptraps.cfg**.This file is a template for SNMP settings, with all traps commented out.
- Export the environment variable DIRX_SNMP=1 in the environment variable configuration file *install_path***/conf/dirxenv.ini**.

For more information, see the chapter **DirX Directory Files** in the *DirX Directory Administration Reference* and the *Guide for CSP Administrators*.

The DirX Directory SNMPv2 implementation comprises only DirX Directory traps.

# 12.5. Monitoring DirX Directory with Nagios

Nagios is widely used to monitor the status of system hardware, applications and services and alert administrators when problems appear and also when they are resolved.

Nagios supports the concept of **plugins** for managing the details of the different resources that can be monitored in a given configuration.A Nagios plugin is a shell script, Perl script or binary that provides input parameters for specifying the targeted object to be checked - for example, how many processes run on a host - and provides input parameters for defining warning or critical levels that the administrator wants the plugin to monitor for the targeted object.

In the Nagios environment, the Nagios plugins interoperate with a main Nagios monitoring server.The monitoring server runs on a host (usually running Linux) and periodically polls the status of defined objects (hosts, services, applications) by invoking specialized plugins to gather the required monitoring data for this object.Each plugin is tailored to the object it monitors.When called by the Nagios server, the plugin evaluates the object’s status and returns it to the Nagios server.The Nagios server then displays this status graphically for each monitored object.

Nagios plugins can also return **performance data** about an object.Performance data is current data that is relevant to monitoring the performance of the object over time.While the status information returned to the Nagios server is volatile - it is replaced on each call to the plugin – performance data is intended to be collected over time and used for long-term trend analysis.The Nagios server saves the returned performance data in a database reserved to the monitored object and can display it in graphical format later on, for example via pnp4nagios.

Nagios provides standard plugins for monitoring network services, such as SMTP, FTP, POP3, and HTTP, and for monitoring host resources, such as processor load and disk

usage.It also supports a simple plugin interface for creating own service-specific monitoring applications and integrating them into the Nagios environment.

DirX Directory provides a set of specialized Nagios plugins that can be used in an existing Nagios environment to monitor the status of DirX Directory service resources and DirX Directory process operations and collect statistics about these items for later performance analysis.

The DirX Directory Nagios plugins allow administrators to monitor important aspects of DirX Directory service operation with Nagios, such as:

- The frequency of DirX Directory database backup operations
- The capacity of DBAM devices configured in a DirX Directory installation
- The operational status of the DBAM database
- The operating mode of the DirX Directory service
- Resources consumed by DirX Directory processes, such as CTX memory and file descriptors
- The responsiveness of DirX Directory processes (the DSA and LDAP server) running in standalone and supplier-consumer replication configurations
- Resource and operational statistics kept by the LDAP server in its MIBs and the changes to these counters over time
- Proper administration of LDIF and shadowing agreements
- Result sizes returned by LDAP search operations and the duration of LDAP client calls, categorized across incremental ranges of sizes and run times

The DirX Directory Nagios plugins provide input parameters for the user to specify warning and critical thresholds to be monitored for these DirX Directory service objects, offering DirX Directory administrators the opportunity to respond to problems detected by the plugins and displayed by the Nagios server before they become severe, and to track their resolution.

The DirX Directory Nagios plugins are delivered as full-function Perl-based executables that can be copied and then extended to address customer-specific DirX Directory monitoring requirements.For details about DirX Directory Nagios plugin configuration, setup, operation and calling syntax, see the *DirX Directory Plugins for Nagios*.For details about the Nagios Core monitoring application, see the See the Nagios documentation.

## 12.6. Monitoring DirX Directory with the Supervisor Scripts

High availability is often a key requirement of directory service deployments.DirX Directory supports both hardware-based and software-based solutions for high availability and failover, including floating master X.500 directory replication in the software layer and clusters and RAID arrays in the hardware layer.

For floating-master replication configurations, DirX Directory includes a software

monitoring and recovery tool called the DirX Directory Supervisor that provides for continuous, periodic monitoring of shadowing-level and network-level connectivity between the DirX Directory service instances in the configuration and automatic recovery from many types of responsiveness problems.The goal of the Supervisor scripts is to ensure that a supplier DirX Directory database is always available in the floating-master configuration.

The DirX Directory Supervisor consists of the **dirxsupervisor** command that executes Tcl- and Perl-based scripts that DirX Directory administrators run on each DirX Directory service in the floating-master configuration.Each Supervisor script instance monitors a pair of DirX Directory service instances (DSA, LDAP server and DirX Directory database): the supplier and one of the consumers.

The scripts perform their responsiveness monitoring and recovery operations on supplier and consumer instances over LDAPv3, DAP/RPC, DSP/DISP and ping (Packet Internet Groper) and log the results of each operation for administrative notification.The following figure illustrates the Supervisor-to-DirX Directory service relationship and the monitoring/recovery protocols used.

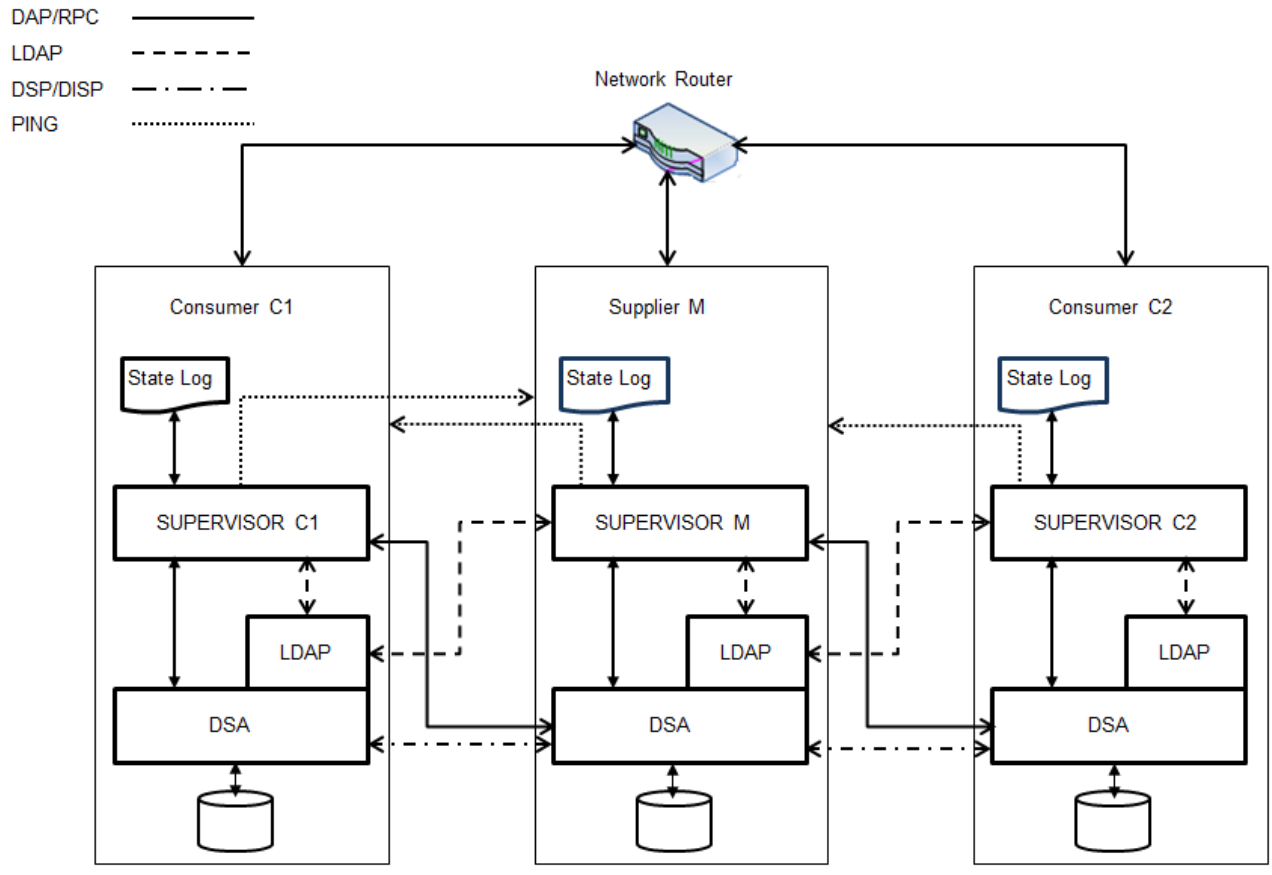



*Figure 35. Supervisor Operation in a Floating-Master Configuration*

The next sections describe the monitoring and recovery operations in more detail and describe how to set up the Supervisor scripts. In the sections that follow, the set of Supervisor scripts is referred to as "the Supervisor".

### 12.6.1. Monitoring Operations

A Supervisor instance runs co-located on every DirX Directory service operating in the floating-master configuration. It monitors shadowing and network-level responsiveness between the local host and one remote instance.

The Supervisor has two modes of operation - supplier or consumer - depending upon the replication role of the co-located DirX Directory DSA. The Supervisor determines the co-located DSA's role on startup and can automatically change its operating mode if the DSA role changes as the result of a **dirxadm sob switch** operation. Only one Supervisor instance at a time can operate in supplier mode.

A Supervisor instance monitors connectivity between a pair of DSAs in two ways:

- By periodically directing asynchronous LDAP operations called "heartbeat operations" to the local DSA and to the remote partner DSA to monitor shadowing responsiveness.
- By periodically sending **ping** commands to a network router (or to the remote partner DSA in the pair, if a router is not part of the configuration), to monitor network responsiveness.

Before each heartbeat operation, the Supervisor checks the co-located DSA's role to make sure that a **dirxadm sob switch** operation has not occurred.

In supplier mode, the heartbeat operation consists of an LDAP bind and update operation to the local DSA that adds a timestamp to the **description** attribute of an LDAP entry that exists in the replicated part of the DIT, and a subsequent read/compare of the test entry on the remote consumer DSA. The DirX Directory administrator specifies the DN to be used for the bind and the target test entry to be modified as part of the Supervisor script setup process.

In consumer mode, the heartbeat operation consists of an LDAP bind and read operation on the local (replicated) test entry's **description** attribute, using the same configured bind DN, and a subsequent read operation of the (master) test entry on the remote supplier DSA.

### 12.6.2. Recovery Operations

The Supervisor attempts to recover from connectivity problems detected by its monitoring operations in two ways:

- By attempting to restart the DirX Directory service a number of times; this number is customizable during Supervisor setup.
- By performing an emergency switch to transfer mastership of the DIT to the partner DSA, if repeated attempts to re-start the service fail.

The emergency switch is not performed until the number of service restart retries is exhausted or a certain network condition is satisfied to allow intermittent or short-term responsiveness problems like network outages to resolve themselves first.

In the event of an emergency switch:

- There is always the possibility of data loss on the new supplier DSA, since it's possible that the consumer DSA's shadowed database copy was not completely synchronized with the old supplier at the time of the emergency switch.
- A total update must be performed to the former supplier DSA once it is up and running again. If synchronous replication is in effect, the DSA on the new supplier automatically performs the total update once the old supplier is running as a consumer. If asynchronous replication is in effect, the DirX Directory administrators must perform the total update manually using **dirxadm sob terminate** and **dirxadm sob establish** operations.

For both emergency and non-emergency switches:

- All LDAP update clients that are directed to the machine that hosts the old supplier may need to be redirected to the machine that hosts the new supplier.
- All LDAP servers must be restarted to update their read-only/read-write status. You can do this by restarting them with different LDAP configuration subentries (the value of the LDAP Read Only Server (LROS) attribute of the LDAP configuration subentry controls the status).

These manual administrative procedures affect the Supervisor's operating parameters, and you will likely need to change the settings in the Supervisor setup script **setup_common_data.tcl** accordingly. For example, you may need to update the Supervisor's **ldapPort** parameter setting to the LDAP port number used on the new supplier's host machine. Note that

The Supervisor cannot recover from problems like a corrupted database; problems like this require administrator intervention. As a result, it is a good idea to examine the Supervisor state logs carefully.

The next sections describe the problems that can be discovered by Supervisor monitoring operations and the general follow-up recovery procedures performed. See the section "Monitoring and Recovery Scenarios" for examples of the recovery procedures performed and the state logging that accompanies them on supplier and consumer sides. any change of the **setup_common_data.tcl** requires a supervisor restart. Also note that it is recommended to apply the same ldap port number on all LDAP servers for monitoring with the supervisor.

#### 12.6.2.1. Recovering from Administrator-Initiated Switches

When a DirX Directory administrator switches the supplier manually, each Supervisor instance detects the DSA role change, automatically switches its role, and logs the incident. See Example 6 in the "Monitoring and Recovery Scenarios" section for an example of this procedure.

#### 12.6.2.2. Recovering from Stopped or Unresponsive Consumers

When the consumer has stopped or is unresponsive:

- Each Supervisor instance detects and logs the incident.
- The consumer-side Supervisor attempts to restart the DirX Directory service.

- The supplier-side Supervisor logs the replication problem for this consumer.

This recovery procedure is effective for consumer restart problems, heartbeat operation timeouts, and LAN outages. See Example 1 in the "Monitoring and Recovery Scenarios" section for an example of this recovery procedure.

#### 12.6.2.3. Recovering from Stopped or Unresponsive Suppliers

When the supplier has stopped or is unresponsive:

- Each Supervisor instance detects and logs the incident.
- The supplier-side Supervisor attempts to restart the DirX Directory service a configurable number of times (the default is 3). If all of these restarts fail, it performs an emergency switch to its partner consumer and exits.
- All remaining Supervisor instances automatically detect and adjust to the role change.

This recovery procedure is effective for supplier restart problems, heartbeat operation timeouts, and LAN outages. See Examples 1 and 2 in the "Monitoring and Recovery Scenarios" section for examples of this recovery procedure.

When the local DirX Directory service is successfully restarted, the DSA starts up as a consumer if an emergency switch was performed and you must manually restart the Supervisor.

The DSA starts up as the supplier again if the cause of its unresponsiveness was a short-duration network outage. In longer-term network outages where an emergency switch has been performed, this can lead to dual mastership errors, discussed in the next section.

#### 12.6.2.4. Recovering from Dual Mastership Errors on Emergency Switches

Dual mastership errors (two supplier DSAs) are usually the result of a long-term local area network outage between the supplier and consumer instances that causes the consumer-side Supervisor to perform an emergency switch to the local DirX Directory service instance, which then runs as the new supplier. The old supplier, meanwhile, will continue to get updates from its locally connected clients, as will the new supplier on the other side of the outage.

When the network returns to service, there are now two supplier instances that claim ownership of the master database in the floating master topology. To recover from this problem, the supplier-mode Supervisors compare the timestamps of the two cooperating DSA tables. Generally, the supplier DSA resulting from the emergency switch has the newer cooperating DSA table, so the DirX Directory service on the instance with the older table is restarted to ensure that the DSA falls back to the consumer role on startup.

The DSA on the new supplier then issues a total update to the old supplier; this means that any updates made to this old supplier’s DirX Directory database during the network outage will be lost. The DSA only performs this automatic total update for synchronous shadowing configurations. See the chapter "Creating a Synchronous Shadow DSA" for an example of this configuration. For asynchronous shadow configurations, DirX Directory administrators must perform the total updates manually using **dirxadm sob terminate** and **sob establish**

operations; see the section "Restoring the Failed Master to the Configuration" in the chapter "Creating a Shadow DSA" for an example.

See Example 3 in the "Monitoring and Recovery Scenarios" section for an example of this recovery procedure.

#### 12.6.2.5. Recovering from Disabled Shadow Operational Bindings

A Supervisor instance running in supplier mode performs a shadowing operational binding (SOB) check as part of its monitoring operation. When it detects a disabled shadowing agreement on the supplier or the partner consumer, it generates a state log that notifies the administrator about the problem. Note that the Supervisor cannot recover from disabled shadowing agreements; the DirX Directory administrator must take action to solve the problem. See Examples 4 and 5 in the section "Monitoring and Recovery Scenarios" for an example of Supervisor operation to detect and notify about disabled SOBs.

### 12.6.3. Setting up the Supervisor

To set up floating master configurations for Supervisor monitoring, on each machine that hosts the DirX Directory service:

- Install the prerequisite software packages
- Deploy the DirX Directory Supervisor scripts
- Establish a Supervisor bind DN and a heartbeat-checking test entry
- Set up an LDAP server for Supervisor monitoring
- Adjust the DirX Directory abbreviation file for handling Supervisor operation
- Update the Supervisor setup script

#### 12.6.3.1. Installing the Prerequisite Software

The DirX Directory Supervisor requires the following software to be installed on each machine in the floating-master configuration:

- The Perl language/interpreter distribution, version 5.16.3 or newer.
- The perl-ldap distribution, which is a collection of Perl modules that provides an object-oriented interface to an LDAP server.

See the **ReadMe.txt** file in the Monitoring installation folder (*install_path**/monitoring*) for instructions on how to set up PERL on the supported DirX Directory platforms.

The Supervisor can be configured to send its state logs in email messages to configurable recipients.

If you plan to use this email notification feature on Windows based DirX Directory installations, you need to obtain the freeware email Windows client binary **bmail.exe** and install it on a Windows machine that is hosting a DirX Directory service instance in the floating-master configuration.

On Linux, the email notification feature uses the built-in **mailx** client.

#### 12.6.3.2. Deploying the Supervisor

By default, the DirX Directory Supervisor scripts are located in the DirX Directory installation path *install_path*/monitoring/supervisor*. Because the Supervisor script settings must be the same for each floating-master instance, we recommend that you set up one copy of the Supervisor scripts on a shared resource that is accessible to all of the machines that host floating-master instances. Otherwise you will need to maintain separate identical setup scripts on each machine. Note, if there is need for using different LDAP port number on the local and remote floating master for the applications, it is recommended to setup separate LDAP servers with identical LDAP port numbers on both servers for the purpose of monitoring with the supervisor. Finally create the supervisor password file. See “Supervisor Password File” in the *DirX Directory Supervisor* for details.

#### 12.6.3.3. Setting up the Supervisor Entries (Supplier Instance Only)

The Supervisor requires an entry to be specified for Supervisor script binds and for heartbeat checking operations. You can use the same entry for both requirements, or you can use two separate entries.

You can create a dedicated Supervisor user entry with any DAP or LDAP client (**dirxcp** or DirX Directory Manager), or you can use an existing user entry, like the default administrator entry. The user entry selected for Supervisor binds:

- Must be an ordinary LDAP entry, for example with object class **inetOrgPerson**.
- Must be an authorized **dirxadm** user; as a result, if you create a dedicated Supervisor user entry, you must be sure to add it to the DirX Directory Administrators (DADM) attribute. See the section **DirX Directory DSA Operational Attributes** in the *DirX Directory Syntaxes and Attributes* for details.
- Must be granted by access control to add, read and update its own entry.

As with the Supervisor user entry, you can create a dedicated heartbeat-checking entry with any DAP or LDAP client, or use an existing one. The entry selected for heartbeat checking operations:

- Must be an ordinary LDAP entry; for example, with object class **inetOrgPerson**.
- Must be located in the area of the DIT that is replicated for shadowing.
- Must have a **description** attribute.
- Must be subject to an AccessControl Item that grants add, read and update operations to the Supervisor user.

Note that if you use the Supervisor user entry as the heartbeat-checking entry, it will need to have a description attribute and be located in the replicated part of the DIT.

#### 12.6.3.4. Setting up the Monitoring LDAP Server (Supplier Instance Only)

The DirX Directory Supervisor scripts require one LDAP server on the supplier instance to be specified for handling heartbeat-checking operations. The selected server must have

the following characteristics:

- It must be a read-write LDAP server (the LROS attribute in the LDAP configuration entry should be **FALSE**)
- It must be set up for local searches only (the LSES attribute in the LDAP configuration subentry should have the values **PreferChaining=F**, **ChainingProhibited=T**, **LocalScope=T**)

You can use **dirxcp modify** commands or DirX Directory Manager (Configuration view -> LDAP configuration subentry -> General tab and Search Service Controls tab) to adjust the LDAP configuration subentry for the LDAP server on the supplier to be used for Supervisor monitoring operations. Remember to restart the modified LDAP server for the new settings to take effect.

See the section "Setting up the LDAP Server" for a discussion of LDAP server operation service controls and how to set them with **dirxcp**. See the section "Extending the My-Company LDAP Server" for examples of how to update existing LDAP configuration subentries and create additional LDAP servers with **dirxcp**.

#### 12.6.3.5. Updating the DirX Directory Abbreviation File

To prepare the DirX Directory abbreviation file for use with the Supervisor, ensure that it contains the **NO_STX_SUFFIX** keyword at the start of the file. See the section **DirX Directory Files** in the *DirX Directory Administration Reference* for details.

#### 12.6.3.6. Updating the Supervisor Setup Script

The final task is to edit the Supervisor setup script **setup_common_data.tcl** and supply the required setup parameter values for the Supervisor and the values of any optional parameters you choose to use. Required parameters include:

- The IP address of a network router in the configuration, or the hostname or IP address of one of the DirX Directory Service instances in the configuration, if there is no router present or if router monitoring is out of monitoring scope.
- The LDAP port and the LDAP configuration subentry of the selected LDAP server selected to handle heartbeat-checking operations. It is assumed that all LDAP servers to be connected are using this LDAP port.
- The distinguished name and the password of the selected Supervisor user entry.
- The language setting of the operating system (controls the language used in **net start dirx** / **net stop dirx** log messages).

See the **setup_common_data.tcl** reference in the *DirX Directory Supervisor* for a description of the required and optional setup parameters and their syntax.

### 12.6.4. Setting up an Email Client (Optional

To use the email notification feature with a running Supervisor, be sure to start it with email notification enabled. The Supervisor core script provides an argument for enabling and disabling email notification and for specifying the severity level of monitoring and recovery

incidents that will trigger email notification. For example, notification of a **dirxadm sob switch** is sent whenever email notification is enabled (by specifying a non-zero value), but notification of a DirX Directory service stop is only sent if the highest level of detail is specified (a value of **3**). See the **dirxsupervisor.tcl** reference page in the *DirX Directory Supervisor* for details.

#### 12.6.4.1. Setup on Windows

The Supervisor can be set up to use the freeware Windows email client **bmail.exe** to send state logs to a Windows email recipient. To enable this feature:

- Install the **bmail.exe** client on a DirX Directory service instance in the floating-master configuration that is running on a Windows platform.
- Make **bmail.exe** available to the Supervisor by copying the executable to the Supervisor's working directory (by default, *install_path**/monitoring/supervisor* or the location on a shared device) or by setting the PATH variable appropriately.
- Supply the following parameters to the Supervisor setup script:
- The address of the email recipient
- The address to use to identify the sender of the email
- The SMTP gateway to use for sending the state log emails

#### 12.6.4.2. Setup on Linux

On Linux platforms, the Supervisor uses the **mailx** command, which is built into the operating system. Separate installation of this email client is not necessary.

When using **mailx**, the only Supervisor-specific data you need to configure is the address of the email recipient.

### 12.6.5. Starting the Supervisor

The Supervisor must be started on each DirX Directory service in the configuration. To start the Supervisor, use the **dirxsupervisor** command. We recommend that you start the Supervisor on the master DirX Directory service first, and then start it on the consumer(s).

The Supervisor core script has several control parameters:

- Whether to allow emergency switch operations in the event of an unresponsive supplier; by default, emergency switches are disabled.
- Whether to recover from any outage problem at all. This running mode only monitors the responsiveness of DirX Directory services and is helpful for testing the setup parameters in advance (simulation mode).
- Whether to send state logs in email messages to a Windows email client; by default, email notification is disabled. Enable it only if you have set up a Windows email client, as described in "Setting up a Windows Email Client".
- Whether to display detailed progress information during script operation; by default, detailed reporting is disabled. Enable it only if you are customizing the Supervisor

scripts and need the details for debugging purposes. It should be disabled when running the Supervisor in production environments.

See the **dirxsupervisor** reference in the *DirX Directory Supervisor* for a detailed syntax description.

### 12.6.6. Stopping the Supervisor

Once started via **dirxsupervisor**, the Supervisor runs continuously. To shut it down, use CTRL/C.

You may need to take the supplier in a floating-master configuration offline for maintenance. In this case, make sure you shut down the Supervisor running on this machine before you shut down the running DirX Directory service.

### 12.6.7. Changing the Monitored Consumer DSA

On startup, the script selects a consumer DSA to monitor based on its position in the cooperating DSA table. It identifies this DSA in its startup output. For example:

```
Processing <0:
TS=DSA1,NA=TCP/IP_IDM!internet=1.2.3.4+port=2,DNS=(HOST=ConsumerDB,SS
LPORT=21201,PLAINPORT=21200)>
        /CN=DirX-DSA-ConsumerDB
        CN=DIRX-DSA-ConsumerDB
        TS=DSA1
        Hostname=ConsumerDB
```

### 12.6.8. Monitoring and Recovery Scenarios

This section provides some examples of problems that can occur in a floating-master configuration, the monitoring and recovery operations performed by supplier and consumer Supervisors and the state logging that accompanies their execution on supplier and consumer sides. For a detailed description of Supervisor state log syntax, see the *DirX Directory Supervisor*.

*Example 1. Stopped DirX Directory Service*

The following Supervisor state logs show the monitoring and recovery operations that occur when the DirX Directory service has stopped on a floating-master instance. This example shows the Supervisor successfully starting the respective DirX Directory service after several connection timeouts occur.

The following state log shows the detection of the problem during a heartbeat checking operation:

13.Dec.2013_16:37:11: POK 1.Heartbeat - general bind/modify problem identified to supplier=tmp1-w2k8-r1 (1)

The next group of messages indicates the attempts to contact the DirX Directory service:

```
13.Dec.2013_16:37:36: POK 1.(getSUK) Local-AE-Show-1 failed with:
                        Error: Directory Server not available.
13.Dec.2013_16:37:47: POK 2.(getSUK) Local-AE-Show-1 failed with:
                        Error: Directory Server not available.
13.Dec.2013_16:37:58: POK 3.(getSUK) Local-AE-Show-1 failed with:
                        Error: Directory Server not available.
13.Dec.2013_16:38:09: NOK (getSUK) Local-AE-Show-1 failed 3 times
with:
                        Error: Directory Server not available.
```

The next state log shows the notification that the Supervisor gives that it will attempt to restart the DirX Directory service:

```
13.Dec.2013_16:38:09: NOK local DSA unresponsive - Restarting DirX
will get invoked now (120)
```

The final state log in this series indicates that the restart was successful.

```
13.Dec.2013_16:38:36:  OK (DirXrestart) 1.DirX-Start and LDAP
connectivity passed
```

*Example 2. Restarting the Supplier DirX Directory Service Continuously Fails*

The following Supervisor state logs show the monitoring and recovery operations that occur when the Supervisor is unable to restart the DirX Directory service on the supplier instance. In this case, the Supervisor on the supplier invokes an emergency switch to the monitored consumer after several LDAP binds and DirX Directory service restarts are refused.

**Supplier Log**

This part of the example shows the state logs from the Supervisor running on the supplier. The logs below identify the problem and the follow-up test operations:

```
13.Dec.2013_16:49:18: POK 2.Heartbeat - general bind/modify
```

```
problem identified to supplier=tmp1-w2k8-r1 (1)
13.Dec.2013_16:49:54: POK 1.(getSUK) Local-AE-Show-1 failed with:
                        Error: Directory Server not available.
13.Dec.2013_16:50:05: POK 2.(getSUK) Local-AE-Show-1 failed with:
                        Error: Directory Server not available.
13.Dec.2013_16:50:16: POK 3.(getSUK) Local-AE-Show-1 failed with:
                        Error: Directory Server not available.
13.Dec.2013_16:50:27: NOK (getSUK) Local-AE-Show-1 failed 3 times
with:
                        Error: Directory Server not available.
```

The next state log shows the notification that the Supervisor will try to restart the DirX Directory service:

```
13.Dec.2013_16:50:27: NOK local DSA unresponsive - Restarting DirX
will get invoked now (120)
```

The next group of state logs shows the failed attempts to restart the service and the notification about the failures:

```
13.Dec.2013_16:50:39: POK (DirXrestart) 1.DirX-Start failed
13.Dec.2013_16:50:51: POK (DirXrestart) 2.DirX-Start failed
13.Dec.2013_16:51:33: POK (DirXrestart) 3.DirX-Start failed
13.Dec.2013_16:51:43: NOK (DirXrestart) The DirX services refuse
to startup 4 times
```

The next state log shows the emergency switch to the consumer:

```
13.Dec.2013_16:51:45:  OK (emergency_switch): Emergency switch to
TMP1-W2K8-R3 passed (EN=0)
```

The final state log in the series indicates that the Supervisor will now shut down, since there is no longer a running DirX Directory service to be monitored:

```
13.Dec.2013_16:51:43:  OK Exit this script, as there is no local
DSA instance to be validated anymore
```

**Consumer Log**

This part of the example shows the state logs from the Supervisor running on the consumer.

The following group of state logs shows the Supervisor detecting the consumer role and successful heartbeat operations to the consumer and supplier:

```
13.Dec.2013_16:48:57: MSG 1.(rolesCHeckNAME) InstanceRole
detected: Shadow
13.Dec.2013_16:49:01:  OK 1.Heartbeat search on consumer=TMP1-
W2K8-R3 PASSED (0)
13.Dec.2013_16:49:09:  OK 1.Heartbeat search on remote
supplier=tmp1-w2k8-r1 PASSED (0)
```

The next group of state logs shows the heartbeat operation passing on the consumer and the connection to the remote supplier failing several times:

```
13.Dec.2013_16:49:19: MSG 1.(rolesCHeckNAME) InstanceRole
detected: Shadow
13.Dec.2013_16:49:23:  OK 1.Heartbeat search on consumer=TMP1-
W2K8-R3 PASSED (0)
13.Dec.2013_16:49:31: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_16:49:47: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_16:50:02: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
```

The next state log shows the notification about the LDAP connection problem to the supplier:

```
13.Dec.2013_16:50:12: NOK General ldap connection problem
identified to supplier-DSA (tmp1-w2k8-r1) 3 times
```

The next state log shows that the consumer Supervisor can successfully ping the router in the configuration even though the LDAP connection request to the supplier is continuously refused:

```
13.Dec.2013_16:50:12:  OK ping to router (192.168.88.130) passed
(CHECK-NO: 1) while ldap connection to tmp1-w2k8-r1 failed
```

The next sequence of state logs shows the consumer Supervisor repeating the same heartbeat operations as shown above, and encountering the same LDAP connection failure:

```
13.Dec.2013_16:50:34: MSG 2.(rolesCHeckNAME) InstanceRole
detected: Shadow
13.Dec.2013_16:50:38:  OK 1.Heartbeat search on consumer=TMP1-
W2K8-R3 PASSED (0)
13.Dec.2013_16:50:46: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_16:51:02: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_16:51:17: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_16:51:27: NOK General ldap connection problem
identified to supplier-DSA (tmp1-w2k8-r1) 3 times
13.Dec.2013_16:51:27:  OK ping to router (192.168.88.130) passed
(CHECK-NO: 2) while ldap connection to tmp1-w2k8-r1 failed
```

The final state log in the series shows the Supervisor detecting that it is now running on the switched supplier:

```
13.Dec.2013_16:51:49: OK  (rolesCheckNAME) SOB-Switch detected:
new Supplier=CN=DIRX-DSA-TMP1-W2K8-R3 (EN=0)
```

*Example 3. Restarting the DirX Directory Service Continuously Fails Due to Network Outage*

The following Supervisor state logs show the monitoring and recovery operations that occur when restarting the DirX Directory service on the supplier fails continuously because of ongoing network connection outages. This example shows the Supervisor on the consumer invoking an emergency switch to itself after several LDAP binds and DirX Directory restarts are refused.

In this situation, the process is as follows:

- The supplier that is unreachable because of the network outage will go on operating locally while its Supervisor periodically reports replication problems to the partner consumer.
- The Supervisor on the consumer also periodically reports a connection problem to the supplier. After several such incidents, the consumer performs an emergency switch to its own service.
- Now there are two shadow masters operating without any interconnection

because of the network outage.

- When the network outage is resolved and both shadow suppliers can interoperate again, the old supplier is automatically restarted as the consumer.

**Supplier Log**

The first group of state logs shows successful heartbeat checking between supplier R1 and consumer R3:

```
13.Dec.2013_17:06:48:  OK 1.Heartbeat operation PASSED to
supplier=tmp1-w2k8-r1 (DSC=1386950804)
13.Dec.2013_17:06:53:  OK 1.Heartbeat compare on consumer=TMP1-
W2K8-R3 PASSED (0)
```

The next set of state logs shows the problem connecting to consumer R3:

```
13.Dec.2013_17:07:06: POK 1.(getSUK) Remote-AE-Show-1 (TMP1-W2K8-
R3) failed with:
                       Error: Directory Server not available.
13.Dec.2013_17:07:16: POK 2.(getSUK) Remote-AE-Show-1 (TMP1-W2K8-
R3) failed with:
                       Error: Directory Server not available.
13.Dec.2013_17:07:26: POK 3.(getSUK) Remote-AE-Show-1 (TMP1-W2K8-
R3) failed with:
                       Error: Directory Server not available.
13.Dec.2013_17:07:36: NOK (getSUK) Remote-AE-Show-1 (TMP1-W2K8-R3)
failed 3 times with:
                       Error: Directory Server not available.
```

The next set of state logs shows the effect of the network outage: R1 continues to identify itself as the supplier, and remote heartbeat operations to R3 continue to fail:

```
13.Dec.2013_17:07:36: MSG (rolesCheckNAME) InstanceRole detected:
Master (on tmp1-w2k8-r1)
        supplierHost:   tmp1-w2k8-r1    CN=DIRX-DSA-TMP1-W2K8-R1
/CN=DIRX-DSA-TMP1-W2K8-R1
        consumerHost:   TMP1-W2K8-R3    CN=DIRX-DSA-TMP1-W2K8-R3
/CN=DIRX-DSA-TMP1-W2K8-R3
13.Dec.2013_17:07:36: MSG IncidentCount=3,
lastIncidentTime=13.Dec.2013_17:07:26
13.Dec.2013_17:07:56:  OK 1.Heartbeat operation PASSED to
```

```
supplier=tmp1-w2k8-r1 (DSC=1386950866)
13.Dec.2013_17:08:00: POK 1.Heartbeat connecting to consumer=TMP1-
W2K8-R3 failed (1)
```

The next series of state logs shows that there is no connection established to the partner R3 as long as the network outage is in effect:

```
13.Dec.2013_17:14:16: MSG Operational binding state on remote
consumer: TMP1-W2K8-R3 (EN=0)
                        Error: Directory Server not available.
13.Dec.2013_17:14:16: NOK replication problem detected in one of
the SOBs - administrative attention/intervention recommended
13.Dec.2013_17:14:16: POK 1.(getSUK) Remote-AE-Show-1 (TMP1-W2K8-
R3) failed with:
                        Error: Directory Server not available.
13.Dec.2013_17:14:26: POK 2.(getSUK) Remote-AE-Show-1 (TMP1-W2K8-
R3) failed with:
                        Error: Directory Server not available.
13.Dec.2013_17:14:36: POK 3.(getSUK) Remote-AE-Show-1 (TMP1-W2K8-
R3) failed with:
                        Error: Directory Server not available.
13.Dec.2013_17:14:46: NOK (getSUK) Remote-AE-Show-1 (TMP1-W2K8-R3)
failed 3 times with:
                        Error: Directory Server not available.
```

The next series of state logs shows what happens when the network is back online: the Supervisor detects two supplier DSAs in the configuration, and then implements the recovery operation for switching its local DSA R1 back to a consumer:

```
13.Dec.2013_17:16:29: MSG (rolesCheckNAME) Two Suppliers-Check:
(SUK (tmp1-w2k8-r1): /CN=DIRX-DSA-TMP1-W2K8-R1)

(SUK (TMP1-W2K8-R3): /CN=DIRX-DSA-TMP1-W2K8-R3)
13.Dec.2013_17:16:29: NOK (rolesCheckNAME) Two Suppliers:
(/CN=DIRX-DSA-TMP1-W2K8-R1) vs. (/CN=DIRX-DSA-TMP1-W2K8-R3) ->
Administrative attention required
13.Dec.2013_17:16:29: MSG (rolesCheckNAME) CDSA-MT of local
supplier: (/CN=DIRX-DSA-TMP1-W2K8-R1,tmp1-w2k8-
r1)=20131213155635.512Z
13.Dec.2013_17:16:29: MSG                CDSA-MT of remote
```

```
supplier: (/CN=DIRX-DSA-TMP1-W2K8-R3,TMP1-W2K8-
R3)=20131213161437.419Z
13.Dec.2013_17:16:29: MSG  ##### Remote supplier (TMP1-W2K8-R3)
keeps the most recent CDSA thus stop local supplier (tmp1-w2k8-r1)
13.Dec.2013_17:16:29: MSG  ##### Manually invoke TOTAL_UPDATE on
remote supplier (TMP1-W2K8-R3), local DirX-supplier will be rest
arted
13.Dec.2013_17:16:29: NOK local DSA unresponsive - Restarting DirX
will get invoked now (400)
13.Dec.2013_17:16:52:  OK (DirXrestart) 1.DirX-Stop passed
13.Dec.2013_17:17:28:  OK (DirXrestart) 1.DirX-Start and LDAP
connectivity passed
13.Dec.2013_17:17:52:  OK 1.Heartbeat operation PASSED to
supplier=tmp1-w2k8-r1 (DSC=1386951468)
13.Dec.2013_17:17:57:  OK 1.Heartbeat compare on consumer=TMP1-
W2K8-R3 PASSED (0)
13.Dec.2013_17:18:20: MSG 1.(rolesCHeckNAME) InstanceRole
detected: Shadow
13.Dec.2013_17:18:25:  OK 1.Heartbeat search on consumer=tmp1-
w2k8-r1 PASSED (0)
13.Dec.2013_17:18:32:  OK 1.Heartbeat search on remote
supplier=TMP1-W2K8-R3 PASSED (0)
```

#### Consumer Log

The first group of state logs shows successful heartbeat checking between supplier R1 and consumer R3:

```
13.Dec.2013_17:06:41: MSG 1.(rolesCHeckNAME) InstanceRole
detected: Shadow
13.Dec.2013_17:06:46:  OK 1.Heartbeat search on consumer=TMP1-
W2K8-R3 PASSED (0)
13.Dec.2013_17:06:53:  OK 1.Heartbeat search on remote
supplier=tmp1-w2k8-r1 PASSED (0)
```

The next set of state logs shows the heartbeat operations failing to the supplier R1:

```
13.Dec.2013_17:07:03: MSG 1.(rolesCHeckNAME) InstanceRole
detected: Shadow
13.Dec.2013_17:07:08:  OK 1.Heartbeat search on consumer=TMP1-
```

```
W2K8-R3 PASSED (0)
13.Dec.2013_17:07:33: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_17:07:56: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_17:08:19: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_17:08:29: NOK General ldap connection problem
identified to supplier-DSA (tmp1-w2k8-r1) 3 times
```

This state log shows that a ping to the router attached to R1 was successful, but the LDAP connection to R1 was refused:

```
13.Dec.2013_17:08:29: OK ping to router (192.168.88.130) passed
(CHECK-NO: 1) while ldap connection to tmp1-w2k8-r1 failed
```

The next set of state logs shows the Supervisor continuing to perform heartbeat operations to the supplier R1, which all fail, and then performing a ping operation to the router, which succeeds:

```
13.Dec.2013_17:08:52: MSG 2.(rolesCHeckNAME) InstanceRole
detected: Shadow
13.Dec.2013_17:08:57:  OK 1.Heartbeat search on consumer=TMP1-
W2K8-R3 PASSED (0)
13.Dec.2013_17:09:25: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_17:09:48: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_17:10:11: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_17:10:21: NOK General ldap connection problem
identified to supplier-DSA (tmp1-w2k8-r1) 3 times
13.Dec.2013_17:10:21:  OK ping to router (192.168.88.130) passed
(CHECK-NO: 2) while ldap connection to tmp1-w2k8-r1 failed
13.Dec.2013_17:10:43: MSG 3.(rolesCHeckNAME) InstanceRole
detected: Shadow
13.Dec.2013_17:10:47:  OK 1.Heartbeat search on consumer=TMP1-
W2K8-R3 PASSED (0)
13.Dec.2013_17:11:13: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
```

```
13.Dec.2013_17:11:36: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_17:12:00: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_17:12:10: NOK General ldap connection problem
identified to supplier-DSA (tmp1-w2k8-r1) 3 times
13.Dec.2013_17:12:10:  OK ping to router (192.168.88.130) passed
(CHECK-NO: 3) while ldap connection to tmp1-w2k8-r1 failed
```

The next set of state logs shows the Supervisor exhausting its maximum operation retry and performing an emergency switch to itself (shown in bold type):

```
13.Dec.2013_17:12:31: MSG 4.(rolesCHeckNAME) InstanceRole
detected: Shadow
13.Dec.2013_17:12:36:  OK 1.Heartbeat search on consumer=TMP1-
W2K8-R3 PASSED (0)
13.Dec.2013_17:13:01: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_17:13:30: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_17:13:53: POK 1.Heartbeat connecting to remote
supplier=tmp1-w2k8-r1 failed (1)
13.Dec.2013_17:14:03: NOK General ldap connection problem
identified to supplier-DSA (tmp1-w2k8-r1) 3 times
13.Dec.2013_17:14:03:  OK ping to router (192.168.88.130) passed
(CHECK-NO: 4) while ldap connection to tmp1-w2k8-r1 failed
13.Dec.2013_17:14:47:  OK (emergency_switch): Emergency switch to
localhost passed (EN=0)
```

The final series of state logs shows consumer R3 now operating as the supplier, with old supplier R1 operating as the consumer:

```
13.Dec.2013_17:19:35:  OK 1.Heartbeat operation PASSED to
supplier=TMP1-W2K8-R3 (DSC=1386951571)
13.Dec.2013_17:19:39:  OK 1.Heartbeat compare on consumer=tmp1-
w2k8-r1 PASSED (0)
13.Dec.2013_17:19:53: MSG (rolesCheckNAME) InstanceRole detected:
Master (on TMP1-W2K8-R3)
        supplierHost:   TMP1-W2K8-R3    CN=DIRX-DSA-TMP1-W2K8-R3
/CN=DIRX-DSA-TMP1-W2K8-R3
```

```
        consumerHost:   tmp1-w2k8-r1    CN=DIRX-DSA-TMP1-W2K8-R1
/CN=DIRX-DSA-TMP1-W2K8-R1
13.Dec.2013_17:19:53: MSG IncidentCount=7,
lastIncidentTime=13.Dec.2013_17:16:21
13.Dec.2013_17:20:07:  OK 1.Heartbeat operation PASSED to
supplier=TMP1-W2K8-R3 (DSC=1386951603)
13.Dec.2013_17:20:11:  OK 1.Heartbeat compare on consumer=tmp1-
w2k8-r1 PASSED (0)
```

*Example 4. Shadow Operational Binding Disabled on Supplier*

The following state logs show the monitoring operations and notifications that occur when the shadow operational binding (SOB) is disabled on the supplier. The example shows the supplier-side Supervisor generating notifications about the problem in state logging messages. The consumer-side Supervisor does not log any notifications about the problem because it does not compare heartbeat values when operating in consumer mode.

In this case, the DirX Directory administrator must take action to recover from the problem.

**Supplier Log**

This set of state logs shows the Supervisor checking the shadowing agreement on the supplier DSA and determining that it is disabled (D=TRUE in US) and generating a state log that indicates the problem (shown in bold type):

```
sob show -supp /CN=DirX-DSA-tmp1-w2k8-r1 -cons /CN=DirX-DSA-tmp1-
w2k8-r3 -agr 88
OBS=COOPERATIVE OBMV={VF=131213134341Z}
AGR={SS={AREA={CP={/},RA={DEF=TRUE}},ATT={DEF=TRUE},KNO={KT=BOTH,E
K=TRUE}}}
US={D=TRUE,NU=1970010100000Z,US={SUS={AC=CHANGED,OU={SID=0,UT=201
31214203753Z,OPMSN=633,AC=FALSE},AU={WS=INVOKED}}}}
RPOL={INI={RMR=5,RTO=600,RI=120,ATU=TRUE},CONS={REPLS=TRUE}}
local_dsaname: /CN=DirX-DSA-tmp1-w2k8-r1
con: /CN=DirX-DSA-tmp1-w2k8-r3
--> SOB=88: D=TRUE (disabled)
```

This set of state logs shows the Supervisor checking the state of the shadowing agreement on the consumer, which shows it is enabled (D=FALSE in US) and generating a state log (shown in bold type) that notifies the administrator about the problem:

```
14.Dec.2013_21:39:01: MSG Operational binding state on remote
consumer: TMP1-W2K8-R3 (EN=0)
                        sob show -supp /CN=DirX-DSA-tmp1-w2k8-r1
-cons /CN=DirX-DSA-tmp1-w2k8-r3 -agr 88
OBS=COOPERATIVE OBMV={VF=131213134341Z}
AGR={SS={AREA={CP={/},RA={DEF=TRUE}},ATT={DEF=TRUE},KNO={KT=BOTH,E
K=TRUE}}}
US={D=FALSE,NU=19700101000000Z,US={CUS={LU=20131214203753Z,CUS=INC
REMENTAL-UPDATE}}}
RPOL={INI={RMR=5,RTO=600,RI=120,ATU=TRUE},CONS={REPLS=TRUE}}
/CN=DirX-DSA-tmp1-w2k8-r1 /CN=DirX-DSA-tmp1-w2k8-r3
14.Dec.2013_21:39:01: NOK replication problem detected in one of
the SOBs - administrative attention/intervention recommended
```

#### Consumer Log

The consumer-side Supervisor does not generate any state logs about the problem because, in consumer-mode operation, it does not run the monitoring operation that detects the problem.

*Example 5. Shadow Operational Binding Disabled on Consumer*

The following state logs show the monitoring operations and notifications that occur when the shadow operational binding (SOB) is disabled on the consumer. The example shows the supplier-side and consumer-side Supervisor generating notifications about the problem in state logging messages.

In this case, the DirX Directory administrator must take action to recover from the problem.

#### Supplier Log

This set of state logs shows the Supervisor checking the shadowing agreement on the supplier DSA and detecting the disabled agreement on the consumer (shown in bold type):

```
sob show -supp /CN=DirX-DSA-tmp1-w2k8-r1 -cons /CN=DirX-DSA-tmp1-
w2k8-r3 -agr 88
OBS=COOPERATIVE OBMV={VF=131213134341Z}
AGR={SS={AREA={CP={/},RA={DEF=TRUE}},ATT={DEF=TRUE},KNO={KT=BOTH,E
K=TRUE}}}
US={D=FALSE,NU=20131214205016Z,US={SUS={AC=TOUCHED,OU={SID=0,UT=20
131214204743Z,OPMSN=636,AC=TRUE},FU={SID=0,UT=20131214204815Z,AC=F
```

```
ALSE,OS=0,MC=1,FR=4002:INACTIVE_AGREEMENT},AU={WS=INVOKED}}}}
RPOL={INI={RMR=5,RTO=600,RI=120,ATU=TRUE},CONS={REPLS=TRUE}}
local_dsaname: /CN=DirX-DSA-tmp1-w2k8-r1
con: /CN=DirX-DSA-tmp1-w2k8-r3
--> SOB=88: FU= (Fault reason)
```

This set of state logs shows the Supervisor checking the state of the shadowing agreement, which shows it is disabled (D=TRUE in US) and generating a state log (shown in bold type) that notifies the administrator about the problem:

```
14.Dec.2013_21:48:43: MSG Operational binding state on remote
consumer: TMP1-W2K8-R3 (EN=0)
                        sob show -supp /CN=DirX-DSA-tmp1-w2k8-r1
-cons /CN=DirX-DSA-tmp1-w2k8-r3 -agr 88OBS=COOPERATIVE
OBMV={VF=131213134341Z}
AGR={SS={AREA={CP={/},RA={DEF=TRUE}},ATT={DEF=TRUE},KNO={KT=BOTH,E
K=TRUE}}}
US={D=TRUE,NU=197001010000000Z,US={CUS={LU=20131214204743Z,CUS=INCR
EMENTAL-UPDATE}}}
RPOL={INI={RMR=5,RTO=600,RI=120,ATU=TRUE},CONS={REPLS=TRUE}}
--> SOB=88: D=TRUE (disabled)
/CN=DirX-DSA-tmp1-w2k8-r1 /CN=DirX-DSA-tmp1-w2k8-r3
14.Dec.2013_21:48:43: NOK replication problem detected in one of
the SOBs - administrative attention/intervention recommended
```

#### Consumer Log

This set of state logs shows the Supervisor indicating the disabled shadowing agreement on the consumer (shown in bold type) and notifying the administrator about the problem (shown in bold type):

```
sob show -supp /CN=DirX-DSA-tmp1-w2k8-r1 -cons /CN=DirX-DSA-tmp1-
w2k8-r3 -agr 88
OBS=COOPERATIVE OBMV={VF=131213134341Z}
AGR={SS={AREA={CP={/},RA={DEF=TRUE}},ATT={DEF=TRUE},KNO={KT=BOTH,E
K=TRUE}}}
US={D=TRUE,NU=197001010000000Z,US={CUS={LU=20131214204743Z,CUS=INCR
EMENTAL-UPDATE}}}
RPOL={INI={RMR=5,RTO=600,RI=120,ATU=TRUE},CONS={REPLS=TRUE}}
--> SOB=88: D=TRUE (disabled) CINS=FALSE
/CN=DirX-DSA-tmp1-w2k8-r1 /CN=DirX-DSA-tmp1-w2k8-r3
```

```
14.Dec.2013_21:48:38: NOK replication problem detected in one of
the SOBs - administrative intervention recommended (xEN=0)
```

*Example 6. SOB Switch to Consumer*

This example illustrates the monitor and recovery operations on a **dirxadm sob switch** to the consumer. In this case, the Supervisor on both supplier and consumer log the incident.

**Supplier Log**

This set of state logs shows the Supervisor performing heartbeat checking operations in supplier-mode:

```
14.Dec.2013_22:00:56:  OK 1.Heartbeat operation PASSED to
supplier=tmp1-w2k8-r1 (DSC=1387054852)
14.Dec.2013_22:01:01:  OK 1.Heartbeat compare on consumer=TMP1-
W2K8-R3 PASSED (0)
```

The next set of state logs shows the Supervisor detecting the DSA role change to consumer:

```
14.Dec.2013_22:01:14: MSG (rolesChecKNAME) InstanceRole detected:
Shadow
14.Dec.2013_22:01:14: MSG Continue proofing ownRole change on
shadow/consumer-side
```

After issuing this state log, the Supervisor changes to consumer-mode operation.

**Consumer Log**

This set of state logs shows the Supervisor performing heartbeat-checking operations in consumer-mode:

```
14.Dec.2013_22:00:50: MSG 1.(rolesCHeckNAME) InstanceRole
detected: Shadow
14.Dec.2013_22:00:54:  OK 1.Heartbeat search on consumer=TMP1-
W2K8-R3 PASSED (0)
14.Dec.2013_22:01:02:  OK 1.Heartbeat search on remote
supplier=tmp1-w2k8-r1 PASSED (0)
```

The next set of state logs shows the Supervisor detecting the DSA role change to supplier:

```
14.Dec.2013_22:01:12: OK  (rolesCheckNAME) SOB-Switch detected:
new Supplier=CN=DIRX-DSA-TMP1-W2K8-R3 (EN=0)
14.Dec.2013_22:01:12: MSG IncidentCount=1,
lastIncidentTime=14.Dec.2013_22:01:12
```

After issuing this state log, the Supervisor changes to supplier-mode operation.

**Consumer Log on Remaining Consumer**

This set of state logs shows the Supervisor running on another consumer performing heartbeat-checking operations in consumer-mode:

```
14.Dec.2013_22:00:58: MSG 1.(rolesCHeckNAME) InstanceRole
detected: Shadow
14.Dec.2013_22:01:03:  OK 1.Heartbeat search on consumer=TMP1-
W2K8-R2 PASSED (0)
14.Dec.2013_22:01:10:  OK 1.Heartbeat search on remote
supplier=tmp1-w2k8-r1 PASSED (0)
```

The next set of messages shows this Supervisor detecting the supplier change during heartbeat checking operations:

```
14.Dec.2013_22:01:20: MSG 1.(rolesCHeckNAME) InstanceRole
detected: Shadow
14.Dec.2013_22:01:25:  OK 1.Heartbeat search on consumer=TMP1-
W2K8-R2 PASSED (0)
14.Dec.2013_22:01:33:  OK 1.Heartbeat search on remote
supplier=TMP1-W2K8-R3 PASSED (0)
```

*Example 7. Normal (no Incident) Logging*

This example shows supplier-side Supervisor and consumer-side Supervisor state logging when no incidents are detected.The example shows that both Supervisors generate periodic state logging about heartbeat operation results.

**Supplier Log**

The following series of state logs shows the heartbeat operations performed by the Supervisor on the local supplier and remote consumer.The messages highlighted in

bold indicate success.

```
14.Dec.2013_22:05:47: MSG (rolesCheckNAME) InstanceRole detected:
Master (on tmp1-w2k8-r1)
        supplierHost:   tmp1-w2k8-r1    CN=DIRX-DSA-TMP1-W2K8-R1
/CN=DIRX-DSA-TMP1-W2K8-R1
        consumerHost:   TMP1-W2K8-R3    CN=DIRX-DSA-TMP1-W2K8-R3
/CN=DIRX-DSA-TMP1-W2K8-R3
14.Dec.2013_22:05:47: MSG IncidentCount=0, lastIncidentTime=--:--
:--
14.Dec.2013_22:06:01:  OK 1.Heartbeat operation PASSED to
supplier=tmp1-w2k8-r1 (DSC=1387055157)
14.Dec.2013_22:06:06:  OK 1.Heartbeat compare on consumer=TMP1-
W2K8-R3 PASSED (0)
14.Dec.2013_22:06:19: MSG (rolesCheckNAME) InstanceRole detected:
Master (on tmp1-w2k8-r1)
        supplierHost:   tmp1-w2k8-r1    CN=DIRX-DSA-TMP1-W2K8-R1
/CN=DIRX-DSA-TMP1-W2K8-R1
        consumerHost:   TMP1-W2K8-R3    CN=DIRX-DSA-TMP1-W2K8-R3
/CN=DIRX-DSA-TMP1-W2K8-R3
14.Dec.2013_22:06:19: MSG IncidentCount=0, lastIncidentTime=--:--
:--
14.Dec.2013_22:06:33:  OK 1.Heartbeat operation PASSED to
supplier=tmp1-w2k8-r1 (DSC=1387055189)
14.Dec.2013_22:06:38:  OK 1.Heartbeat compare on consumer=TMP1-
W2K8-R3 PASSED (0)
14.Dec.2013_22:06:51: MSG (rolesCheckNAME) InstanceRole detected:
Master (on tmp1-w2k8-r1)
        supplierHost:   tmp1-w2k8-r1    CN=DIRX-DSA-TMP1-W2K8-R1
/CN=DIRX-DSA-TMP1-W2K8-R1
        consumerHost:   TMP1-W2K8-R3    CN=DIRX-DSA-TMP1-W2K8-R3
/CN=DIRX-DSA-TMP1-W2K8-R3
14.Dec.2013_22:06:51: MSG IncidentCount=0, lastIncidentTime=--:--
:--
14.Dec.2013_22:07:05:  OK 1.Heartbeat operation PASSED to
supplier=tmp1-w2k8-r1 (DSC=1387055221)
14.Dec.2013_22:07:10:  OK 1.Heartbeat compare on consumer=TMP1-
W2K8-R3 PASSED (0)
```

**Consumer Log**

The following group of state logs shows the heartbeat checking operations performed by the consumer Supervisor on the local consumer and remote supplier.The messages highlighted in bold type indicate success.

```
14.Dec.2013_22:05:38: MSG 1.(rolesCHeckNAME) InstanceRole
detected: Shadow
14.Dec.2013_22:05:43:  OK 1.Heartbeat search on consumer=TMP1-
W2K8-R2 PASSED (0)
14.Dec.2013_22:05:50:  OK 1.Heartbeat search on remote
supplier=tmp1-w2k8-r1 PASSED (0)

14.Dec.2013_22:06:00: MSG 1.(rolesCHeckNAME) InstanceRole
detected: Shadow
14.Dec.2013_22:06:05:  OK 1.Heartbeat search on consumer=TMP1-
W2K8-R2 PASSED (0)
14.Dec.2013_22:06:12:  OK 1.Heartbeat search on remote
supplier=tmp1-w2k8-r1 PASSED (0)

14.Dec.2013_22:06:22: MSG 1.(rolesCHeckNAME) InstanceRole
detected: Shadow
14.Dec.2013_22:06:26:  OK 1.Heartbeat search on consumer=TMP1-
W2K8-R2 PASSED (0)
14.Dec.2013_22:06:34:  OK 1.Heartbeat search on remote
supplier=tmp1-w2k8-r1 PASSED (0)
```

## 12.7. Using DSA Auditing

Auditing is another way to observe the DSA, but for billing and accounting management rather than for processing management.When auditing is enabled, the DirX Directory auditing service writes information about the DSA operations that occur into a binary audit log file.Each logged DSA operation is stored as one record in the audit log file.Information recorded about the operation includes:

- The session ID, which is used to distinguish between different connections
- The time the operation started, and the time it ended
- The duration of the operation
- The protocol used for the operation (DAP, DSP, DISP, or local)
- The type of operation performed; for example, a bind, a search, or a remove
- Operation-specific information, such as the IP address of the initiator and the

authentication method used, if the operation was a bind, or the target object if the operation was a remove, or the target object, filter information, and other search operation specific information if the operation was a search

- The result of the operation (success, or failure with error code information, and for search results the size of the result pdu)

Auditing is most often used when a company offers its DirX Directory service to third party customers. The DirX Directory administrators for the company use the DirX Directory auditing tools to track the DSA operations initiated by the third party customers; the company that owns the DirX Directory services can then bill these customers for the number of DSA operations they make and the length of time the operations run.

The tools for managing DirX Directory auditing are:

- The **dirxadm audit** object and its operations, which is used to enable, configure, and manage the auditing of DSA operations
- The **dirxauddecode** program, which is used to convert the binary format audit log file generated by DirX Directory auditing into a human-readable or program-readable format.
- The **dirxaudstatistics** program, which is used to evaluate multiple binary audit log files.
- The LDAP extended operations **dsa_audit_evaluate**, **dsa_audit_enable**, **dsa_audit_disable**, **dsa_audit_info** for managing DSA auditing available through the **dirxextop** command (see the **dirxextop** section in the *DirX Directory Administration Reference* for details)
- DirX Directory Manager's Monitor view (see the *DirX Directory Manager Guide* for details)

The next sections describe the tasks associated with auditing and how you perform them with the DirX Directory auditing tool **dirxadm audit**.

### 12.7.1. Enabling and Disabling Audit Logging

Auditing is disabled by default. To enable auditing, use the **dirxadm** command:

**audit modify -status on**

This command enables auditing immediately. Alternatively, you can use the LDAP extended operation **dsa_audit_enable** to enable auditing.

To disable auditing, use the **dirxadm** command:

**audit modify -status off**

Alternatively, you can use the LDAP extended operation **dsa_audit_disable** to disable auditing.

## 12.7.2. Selecting the Level of Auditing

Once it is enabled, the DirX Directory auditing service, by default, writes a record of every operation the DSA performs to the audit log file. You can restrict audit logging to certain categories of DSA operations by using the **dirxadm** command:

**audit modify -level** *level*

where *level* is one or more of the following keywords:

***ENTRY***
: Use the **ENTRY** keyword to audit only those DSA operations that affect entries.

***ATTR***
: Use the **ATTR** keyword to audit only those DSA operations that affect attributes.

***ATTRVAL***
: Use the **ATTRVAL** keyword to audit only those DSA operations that affect attributes and values.

Using the **-level** option to log only certain types of transactions can help you to minimize the size of the audit log file.

## 12.7.3. Managing the Audit Log File

Managing the audit log file consists of three tasks:

- Establishing the audit log file's size
- Choosing how to handle audit log file overflow
- Saving the data in the audit log file

#### 12.7.3.1. Establishing the Size of the Audit Log File

By default, the audit log file will store a maximum of 50000 records of audit data. You can change this maximum size with the **dirxadm** command

**audit modify -size** *size*

where *size* is an integer that specifies the maximum amount of data in bytes you want the audit log file to store. Use **0** to specify an unlimited maximum size. If you select an unlimited size for the audit log file, be sure to keep track of the audit log file's size and empty it periodically.

#### 12.7.3.2. Handling Audit Log File Overflow

Administrators can choose one of three different strategies for handling a full audit log file. To set the strategy, use the **dirxadm** command:

**audit modify -overflow** *strategy*

where *strategy* is one of the keywords:

***STOP***

Use the **STOP** keyword to direct DirX Directory auditing to stop automatically when the audit log file is full.

***STOP-DSA***

Use the **STOP-DSA** keyword to direct the DSA to shut down automatically when the audit log file is full.

***WRAP***

Use the **WRAP** keyword to direct the DirX Directory auditing service to write overflow data to the top of the audit log file, overwriting any existing data there, when the audit log file is full.

***MOVE***

Use the **MOVE** keyword to direct the DirX Directory auditing service to close and rename the audit file currently in use and to create a new empty audit log file, when the audit log file is full.

#### 12.7.3.3. Saving the Data in the Audit Log File

The audit log file is located in *install_path*/server/audit/audit.log*. You can't change this audit log file name and location, nor is there any way to move the data in the audit log file automatically to another location when it becomes full. Instead, you must explicitly move the data out of the audit log file and into another file using the **dirxadm** command:

**audit modify -move**

If you use this command, **dirxadm** moves the data in the audit log file into a new file in the same directory, which it names in the format:

**audit.log.***date_nnn*

where date is where *date* is the current UTC time in ZULU format in the form YYYYMMDDhhmmss**Z** and *nnn* is a counter from 000 to 999. For example, **audit.log.200105101045l1Z_001** is the filename for an audit log file created on 10 May 2001 at 10:45:11 hours.

To select your own pathname for a saved audit log file, use the **dirxadm** command:

**audit modify -move -destination** *filename*

where *filename* is the complete pathname you give to the saved audit log file.

When you empty the **audit.log** file by moving its contents to another file, the auditing service recreates the **audit.log** file and starts writing data into it again.

# 12.8. Using LDAP Auditing

The LDAP server provides its own auditing system, which is independent from the DSA audit.It is disabled by default, but can be enabled at any time to write information about processed LDAP requests to a file.DirX Directory LDAP auditing includes the following features:

- Configurable audit detail level to trace either full request parameters or just a minimum subset.
- Configurable filters for operation types
- Configurable log file size with automatic overflow handling
- Enabling/disabling at runtime
- Timed start/stop
- Data encryption
- Session tracking

The next sections describe how to set up and manage LDAP auditing.

## 12.8.1. Creating the LDAP Audit Subentry

Audit behavior is controlled by an extra audit subentry that contains attributes that control the default audit operations. If you want to configure LDAP auditing, you must add this subentry to the database. The DirX Directory installation provides a **dirxcp** Tcl script that installs this subentry (CN=ldapAudit) with default settings. See the file:

*install_path***/scripts/stand_alone/default/ldap_auditcfg.cp**

If an audit subentry with the common name **ldapAudit** is present in the database, the setting from this subentry is used to control the audit. If the name of the subentry is to be different, you must add the common name (for example, **ldapAudit2**) as a link to the attribute LACCN (ldapAuditCfgSubschemaCN) in the LDAP configuration subentry.

## 12.8.2. Enabling LDAP Auditing

There are two ways to enable auditing:

- At startup time
- At runtime

To enable LDAP audit at startup, an audit subentry must exist in the database, the link to this subentry bust be filled correctly in the LDAP configuration subentry, and the attribute LAON (ldapAuditOn) must be set to TRUE in the audit subentry.

To enable audit during runtime, you can use the following **dirxadm** commands:

**dirxadm ldap binding**
**dirxadm ldap audit -start**

After these commands complete successfully, LDAP auditing is turned on and runs with the default settings.

You can also use DirX Directory Manager and **dirxextop** to enable audit during runtime.

### 12.8.3. Viewing the Current Audit Settings (dirxadm)

To retrieve the current audit settings, use the command:

**dirxadm ldap audit -info**

This command displays the active audit parameters. Here is an example:

```
==================== LDAP Audit Info  ===========
Hostname           : MYHOST01
Current Local Time: Fri Apr 12 13:10:09 2002
Status             : ON
Destination File   : C:\Program
Files\DirX\Directory\Ldap\audit\ldapConfiguration\audit.log
File Size Limit    : 50000 records
Overflow Policy    : wrapAround
Detail Level       : max
Value Limit        : 128
Op Selection       : all
Buffer Size        : 0
Start Time         : none
Stop Time          : none
Encryption         : none
```

These settings reflect the audit subentry attributes. See the *DirX Directory Syntaxes and Attributes* for a more detailed explanation of these attributes.

You can also use DirX Directory Manager and **dirxextop** to retrieve the current audit settings.

### 12.8.4. Stopping and Restarting LDAP Auditing (dirxadm)

Auditing can be stopped at any time by issuing the command:

**dirxadm ldap audit -stop**

To re-enable it, issue the command:

**dirxadm ldap audit -start**

Note that you can also use DirX Directory Manager and **dirxextop** to stop and re-enable audit during runtime.

### 12.8.5. Using Session Tracking

If an LDAP server receives a request it often does not have any information about the originating user. The following figure illustrates this fact:

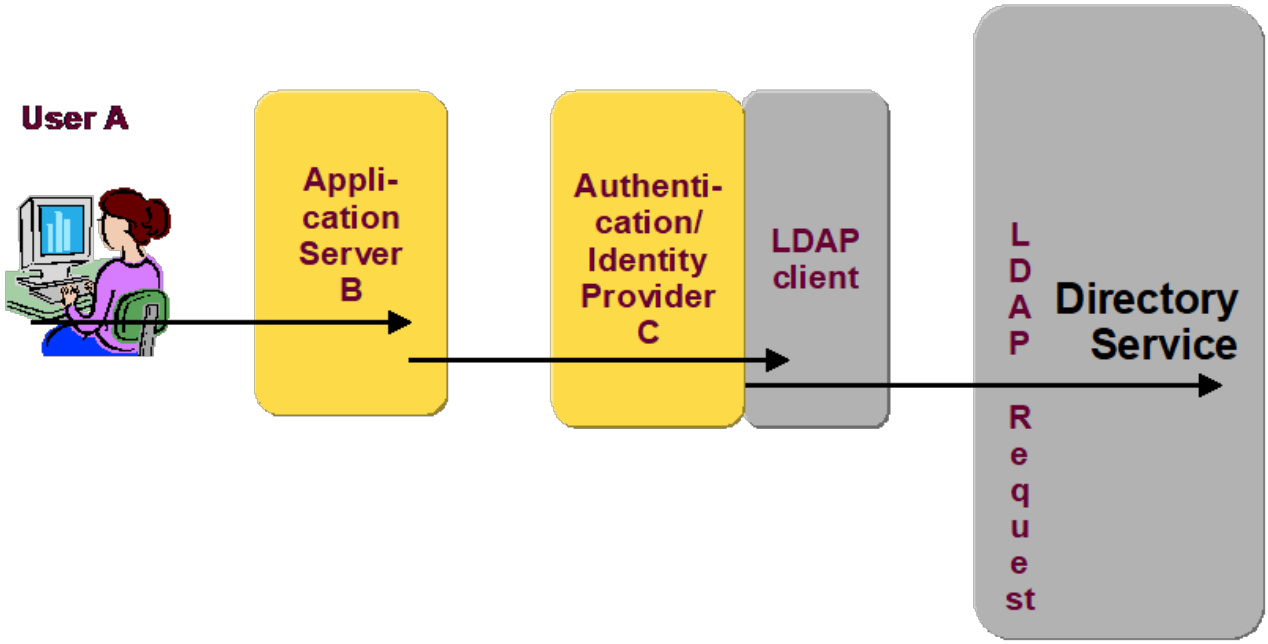



*Figure 36. Motivation for Session Tracking*

In the figure above, user A sends a request to application server B, for example, searching or modifying an entry. Before application server B performs user A's request, it first checks the user A's identity and reads user A's profile through authentication / identity provider C. To fulfill the request of application server B, the authentication / identity provider C generates one or more LDAP requests to the directory service. If authentication / identity provider C does not include the LDAP session tracking control describing user A or application server B in the LDAP requests, the directory service only knows the IP address and the credentials of authentication / identity provider C, and an administrator is unable to track for example user A's interactions.

The LDAP session tracking control enables the LDAP client to provide information about the originating user to the LDAP server. The LDAP session tracking control provides the following information:

- The sessionSourceIP - The IP address of the originating LDAP client application with a maximum length of 128 characters.
- The sessionSourceName - Describes the originating system with a maximum length of 65536 bytes.The value might be for example a hostname, a distinguished name, or a web service address.
- The formatOID - Describes the semantics of the sessionSourceName and the sessionTrackingIdentifier with a maximum length of 1024 bytes.

- The sessionTrackingIdentifier - The tracking identifier, for example, a username that has already been authenticated by the application that is generating the session, with an unlimited length (but should not be longer than 65536 characters).

If enabled, the DirX Directory LDAP server writes this information to the LDAP audit log file.Use the attributes LDAP Audit Session Tracking Enabled and LDAP Audit Session Tracking InfoLen of the LDAP Audit configuration subentry to enable and control logging of the LDAP session tracking control.(See the section “Attributes for LDAP Server Audit Configuration” in the *DirX Directory Syntaxes and Attributes* for details.)

See the Internet-Draft “LDAP Session Tracking Control” (draft-wahl-ldap-session-03.txt) for details about session tracking.

## 12.9. Evaluating the Audit Log File

Both the LDAP server and DSA write the audit log file as a binary file that must be converted to another format in order to use it.Use the **dirxauddecode** program to convert the audit log file from binary format to one of the following formats:

- ASCII format—select ASCII format if you want to be able to read the file
- “Exported” format—select exported format to generate an intermediate binary record format that an accounting or billing application can process

To run **dirxauddecode** on a Windows system:

1. Click **Start**.
2. Search for **Command Prompt**, and then click it.
3. For DSA auditing, change directory to *install_path***server\audit**. For LDAP auditing, change to *install_path***ldap\audit\ldapConfiguration**
4. Enter the **dirxauddecode** command, for example:

   **dirxauddecode -i audit.log.20010510104511Z_001 -a auditlog.txt**

See the **dirxauddecode** section in the *DirX Directory Administration Reference* for a complete description of **dirxauddecode** command line syntax.The section in the *DirX Directory Administration Reference* that describes the **dirxadm audit** object shows a sample ASCII-output audit log file.

Use the **dirxaudstatistics** program to evaluate multiple binary audit log files.See the **dirxaudstatistics** section in the *DirX Directory Administration Reference* for a complete description of **dirxaudstatistics** command line syntax.

You can also use DirX Directory Manager and **dirxextop** to evaluate audit settings.

# 12.10. Using DirX Directory Diagnostic Logging

DirX Directory supports the logging of two types of diagnostic messages:

- Exception messages, which are ASCII-formatted messages that are human-readable
- Trace messages, which are binary formatted messages that must be decoded

DirX Directory provides tools to log trace and exception messages for clients; that is, DirX Directory applications such as **dirxadm**, **dirxcp**, and the LDAP server, and servers; that is, the DSA.The DirX Directory tools for diagnostic logging are:

- The DirX Directory client, server, and tools logging configuration files, which control the level and type of trace and exception logging
- The **dirxadm sys log** and **sys nolog** commands, which you use to switch between levels of diagnostic logging
- The **dirxadm ldap log** and **ldap nolog** commands, which you use to switch between levels of diagnostic logging
- The **dirxdumplog** program, which decodes trace log files so that you can read them
- The DirX Directory client log directory, which is the default repository for logged client exception and trace messages
- The DirX Directory server log directory , which is the default repository for logged server exception and trace messages
- The DirX Directory LDAP server log directory , which is the default repository for logged LDAP server exception and trace messages
- The DirX Directory tools log directory , which is the default repository for logged tools exception and trace messages

Although diagnostic logging is primarily carried out by programmers, DirX Directory administrators may want to set up diagnostic logging and then pass the log files on to programmers for analysis in the event of a serious problem.The next sections provide a summary of the diagnostic logging tools and how to use them.See the *DirX Directory Administration Reference* for complete details about these tools.

# 12.11. Using the Logging Configuration Files

DirX Directory supplies:

- A set of client logging configuration files for the DirX Directory applications (clients) in the client directory *install_path*/client/conf*
- A set of server logging configuration files for the DSA (server) in the server directory *install_path*/server/conf*
- A set of LDAP server logging configuration files for the LDAP server in the LDAP server directory *install_path*/ldap/conf*.
- A set of tools logging configuration files for the tools like **dirxload** in the tools directory *install_path*/tools/conf*

The purpose of two of these files is to specify:

- The DirX Directory client and server subcomponents for which to capture trace messages and the debug level of the traces to log
- The severity of exceptions to log and how and where to log them

One of the files (**dirxlog.on**) specifies the level of trace and exception logging that should occur in the “on” state.The other file (**dirxlog.off**) specifies the level of trace and exception logging that should occur in the “off” state.A third file (**dirxlog.cfg**) is used as the link to the “on” and “off” files.The first two files thus represent the “on/off” positions for the type and level of diagnostic logging performed for client and server, while the link file is used to switch between them.

The logging configuration files delivered with DirX Directory specify default “on” and “off” levels for client and server diagnostic logging.The DirX Directory default “on” file for client and server diagnostic logging enables a default level of trace and exception logging.The client and server “off” files completely turn off trace logging, but keep the default exception logging enabled.DirX Directory is delivered so that the “off” levels are in force.You can tailor the client and server logging configuration files to specify your own “on” and “off” levels of diagnostic logging for client and server.See the “DirX Directory Files” chapter in the *DirX Directory Administration Reference* for complete details about the format of these files.

# 12.12. Enabling and Disabling Diagnostic Logging

DirX Directory clients perform diagnostic logging according to the settings stored in the client configuration file *install_path***/client/conf/dirxlog.cfg**.In order to switch between “on” and “off” levels of diagnostic logging, you must explicitly copy the information in the **dirxlog.on** and **dirxlog.off** files to the **dirxlog.cfg** file.

The DirX Directory DSA performs diagnostic logging according to the settings stored in the server configuration file *install_path***/server/conf/dirxlog.cfg**.To switch between “on” and “off” diagnostic logging settings for the DirX Directory DSA, you use the **dirxadm sys log** and **sys nolog** commands.When you issue the **dirxadm sys log** command, the **dirxadm** program reads the logging level settings stored in the file *install_path***/server/conf/dirxlog.on** and sends them to the DSA, which then logs according to the settings.When you issue the **dirxadm sys nolog** command, **dirxadm** reads the settings stored in the file *install_path***/server/conf/dirxlog.off** and sends them to the DSA, which then logs according to the settings.When a DSA is restarted, it logs according to the settings it finds in the **dirxlog.cfg** file.

The DirX Directory LDAP server performs diagnostic logging according to the settings stored in the server configuration file *install_path***/ldap/conf/dirxlog.cfg**.To switch between “on” and “off” diagnostic logging settings for the DirX Directory LDAP server, you use the **dirxadm ldap log** and **ldap nolog** commands.When you issue the **dirxadm ldap log** command, the **dirxadm** program reads the logging level settings stored in the file *install_path***/ldap/conf/dirxlog.on** and sends them to the LDAP server, which then logs according to the settings.When you issue the **dirxadm ldap nolog** command, **dirxadm** reads the settings stored in the file *install_path***/ldap/conf/dirxlog.off** and sends them to the LDAP server, which then logs according to the settings.When an LDAP server is

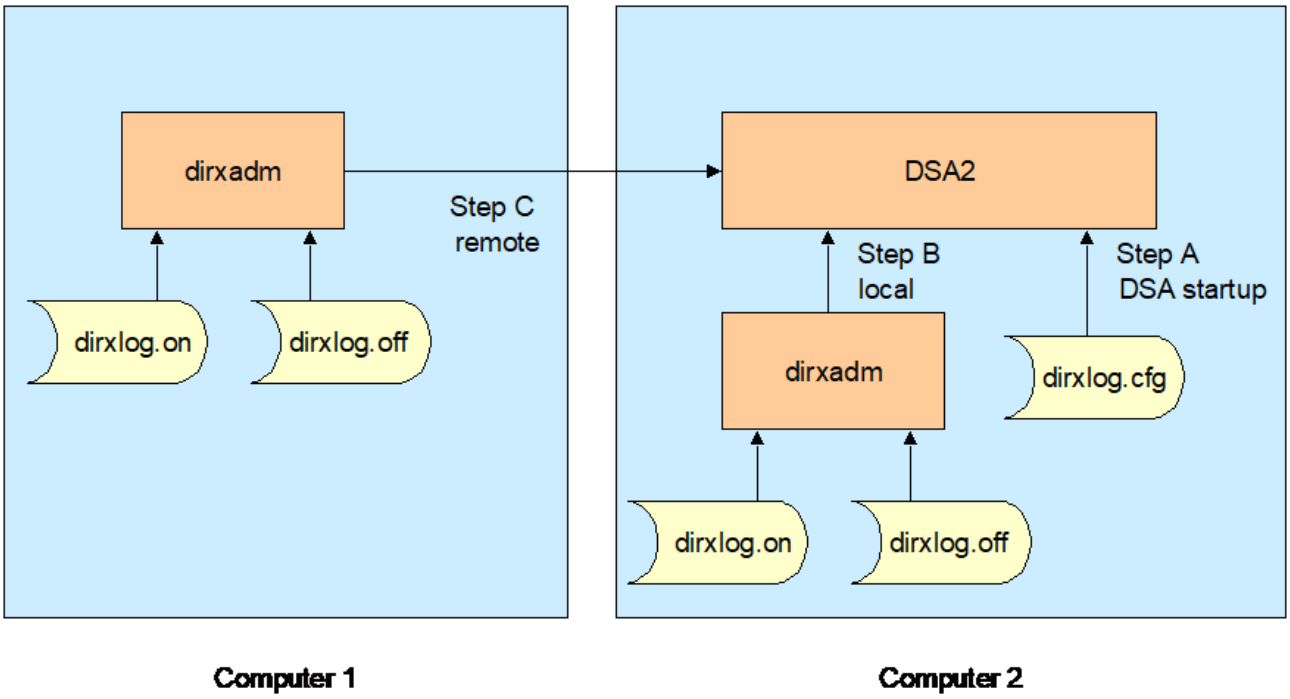
restarted, it logs according to the settings it finds in the **dirxlog.cfg** file.

DirX Directory tools like **dirxload** perform diagnostic logging according to the settings stored in the client configuration file *install_path***/tools/conf/dirxlog.cfg**.In order to switch between “on” and “off” levels of diagnostic logging, you must explicitly copy the information in the **dirxlog.on** and **dirxlog.off** files to the **dirxlog.cfg** file.

You can override this process by using the environment variable **DIRX_LOGCFG_FILE** to establish your own server logging configuration file.When this environment variable is set, all processes perform diagnostic logging using the settings contained in the file specified by **DIRX_LOGCFG_FILE**.The **dirxadm sys log** command and the **dirxadm ldap log** command send the settings in the logging configuration file specified in **DIRX_LOGCFG_FILE** to the DSA or LDAP server, while **dirxadm sys nolog** sends the settings in *install_path***/server/conf/dirxlog.off** to the DSA, and **dirxadm ldap nolog** sends the settings in *install_path***/ldap/conf/dirxlog.off** to the LDAP server.

The following figure illustrates how the logging configuration files are used depending on how a DirX Directory administrator initiates logging.

*Use of Logging Configuration Files*




As illustrated in the figure:

- When a DirX Directory administrator starts the DirX Directory service on computer 2, for example, when he boots the computer or sends a **dirxadm sys start** command, DSA 2 reads the values for logging configuration from the file *install_path***/server/conf/dirxlog.cfg** on computer 2. This procedure is shown in step A.
- When a DirX Directory administrator uses **dirxadm** on computer 2 to turn on logging (the local case) the values saved in the file *install_path***/server/conf/dirxlog.on** on computer 2 are read and sent to DSA 2. DSA 2 then logs according the values sent but does not save them in the file *install_path***/server/conf/dirxlog.cfg** on computer 2. If an

administrator stops DSA 2 and starts it again for any reason (for example, a reboot of computer 2), DSA 2 logs according the values of *install_path***/server/conf/dirxlog.cfg** on computer 2 and not according the values sent the last time by one of the administration tools. This procedure is shown in step B.

- When a DirX Directory administrator uses **dirxadm** on computer 1 to turn on logging (the remote case), the values saved in the file *install_path***/server/conf/dirxlog.on** on computer 1 are read and sent to DSA 2. DSA 2 then logs according the values sent, but does not save them in the *install_path***/server/conf/dirxlog.cfg** file on computer 2. If the administrator stops DSA 2 and starts it again for any reason (for example, a reboot of computer 2) DSA 2 logs according the values of *install_path/***server/conf/dirxlog.cfg** on computer 2 and not according the values sent the last time by one of the administration tools. This procedure is shown in step C.

The same procedures are performed when a DirX Directory administrator uses the administration tools to turn off logging, except that the values saved in the file *install_path***/server/conf/dirxlog.off** are used.

If DSA 2 should always use a particular set of logging values, the administrator must edit the logging configuration file *install_path***/server/conf/dirxlog.cfg** on computer 2 to contain these values.

The same procedures are performed for an LDAP server when a DirX Directory administrator uses the **dirxadm** commands **ldap log on** or **ldap log off**.In this case, the **dirxlog.*** files for the LDAP server that are saved in the directory *install_path***/ldap/conf** are used.

When a DirX Directory administrator sets the environment variable **DIRX_LOGCFG_FILE**, there is only one logging configuration file used for the DSA and LDAP server.The administrator can set this environment variable on computer 1 and on computer 2.

Refer to the *DirX Directory Administration Reference* chapters entitled "File Locations", "DirX Directory Environment Variables", and the section "Logging Configuration Files" in the chapter "DirX Directory Files" for complete details.

# 12.13. Locating and Reading the Diagnostic Log Files

The client and server logging configuration files supplied with DirX Directory also specify default locations to which client and server exception and trace logs are written.

On Linux systems, exception messages from clients are written by default to the client log directory *install_path***/client/log**, exception messages from the DSA are written by default to the server log directory *install_path***/server/log**, exception messages from the LDAP server are written by default to the LDAP server log directory *install_path***/ldap/log**, and exception messages from tools like **dirxload** are written by default to the tools log directory *install_path***/tools/log**.Exception messages are separated into log files of less serious events (named **USR***processID* by default) and log files of exceptions (named **EXC***processID* by default).It is possible to administer DirX Directory and the Linux systems to use the system log daemon additionally.(See the section “syslog Configuration File” in the chapter “DirX Directory Files” in the *DirX Directory Administration Reference* for details.)

On Windows systems, exception messages from clients, servers, and tools are stored in the Windows event log, which you view with the Event Viewer.

Trace log messages are also written by default to the *install_path***/client/log**, *install_path***/server/log**, *install_path***/ldap/log**, and *install_path***/tools/log** directories but are written to a binary file (named **LOG***processID* by default).You can use the logging configuration files to specify your own log file pathnames to which trace and exception messages are written rather than using the defaults.

The exception log files are human-readable ASCII files.

Trace log files are not human-readable.In order to read a trace log file, you must first process it with the **dirxdumplog** program.The section on **dirxdumplog** in the *DirX Directory Administration Reference* describes the **dirxdumplog** program syntax and provides an example of the program's output.

# 12.14. Checking Database Consistency

You can use the **dbamverify** command-line tool on a regular basis to check the consistency of the DirX Directory DBAM database.The **dbamverify** command can check:

- The referential integrity of all attribute value blocks and tree blocks
- The consistency of all directory-specific entries (DSEs) – each DSE is read in the context of this check and is verified with respect to DBAM level consistency, linkage of the entry in the tree, and ASN.1 encoding, schema and index consistency of its attributes
- Attribute value index bit strings
- Subordinate index bit strings

Specifying no option directs **dbamverify** to perform all consistency tests.The command returns an exit code of **0** on success or a **1** if it encountered an error.The text of the error message is displayed on **stderr**.

Because **dbamverify** locks the database for update operations while it runs, it is recommended that you take a backup of the database with **dirxbackup** and then run **dbamverify** on this archive off-line.The section "Managing Backups" provides an example of this approach.

For details on **dbamverify** and **dirxbackup** command-line syntax, see the *DirX Directory Administration Reference*.

# 12.15. Managing Backups

Saving backups of the directory's database is one of the most important tasks of DirX Directory operation.Use only the **dirxbackup** tools for this purpose.Using non-DirX tools that access DBAM database files or devices can produce inconsistencies in the database and / or in the dirxbackup archive file.

You must use the **dirxbackup** command to save and restore a database or verify the consistency of a database archive that you created with **dirxbackup**.Saving a directory

database does **NOT** require setting the directory in read-only mode, so it does not interfere with normal operation.

The **dirxbackup** command supports both full and delta backups.To restore a saved database, first restore the full backup archive and then restore all delta backups in the correct sequence.For details on **dirxbackup** command-line syntax, see the *DirX Directory Administration Reference*.

Here is a sample backup and database consistency-checking scenario:

- Create a full backup every 24 hours - for example, at 12:00 pm - by creating a scheduled job that performs the following **dirxbackup** command:

  `dirxbackup –S dirxDB.`*`date_Time`*

- Create delta backups every *n* hours; for example, every four hours. The actual frequency of delta backups depends on your requirements. To achieve this, create a scheduled job that performs the following **dirxbackup** command:

  `dirxbackup –Sd dirxDB.`*`date_Time`*`.delta.`*`number`*

- Check the archives for internal integrity and completeness by performing:

  `dirxbackup –T dirxDB.`*`date_Time`* `dirxDB.`*`date_Time.number`*

- If the checks on the archives perform successfully, perform the DBAM database consistency check on the archives with the **dbamverify** command.Specify the full backup archive first, followed by the delta backup archives in sequence, for example:

  ```
  dbamverify dirxDB.date_Time dirxDB.date_Time.1 \
  dirxDB.date_Time.2 …
  ```

# 12.16. Exceeding Database Limits

The following errors indicate that your DBAM database limits have been exceeded:

- **DBAM_NOFREEENTRYFOUND 11105**
  This error indicates that there are no more free blocks in a DBAM device.Use the **dbamdevinfo** command-line tool (see the **dbamdevinfo** section in the *DirX Directory Administration Reference*) to find out which DBAM device is full.
- **DBAM_IMPLLIMITEXCEEDED 11102**
  This error indicates that a DBAM design limit is exceeded.For example, if the maximum number of child entries below a parent entry is exceeded, this error is logged.

If one of the errors above is logged, extend your database as described in the chapter titled “Extending the Directory Database” in the *Disc Dimensioning Guide*.

# 13. Understanding External Authentication

The DirX Directory service can use external services such as Windows or RACF to authenticate the users that perform a bind to the directory. In external authentication, the DirX Directory service forwards an LDAPv3 bind request performed with simple bind credentials (distinguished name (DN) and a password) to an external authentication service via the external service's protocol, for example, via the Kerberos protocol used in Windows domains. This action links the DSA to the external service's login facility, which performs the client authentication procedure. On success, a mapping procedure determines the authorization identity used for access control decisions in subsequent directory operations. The following figure illustrates the process.

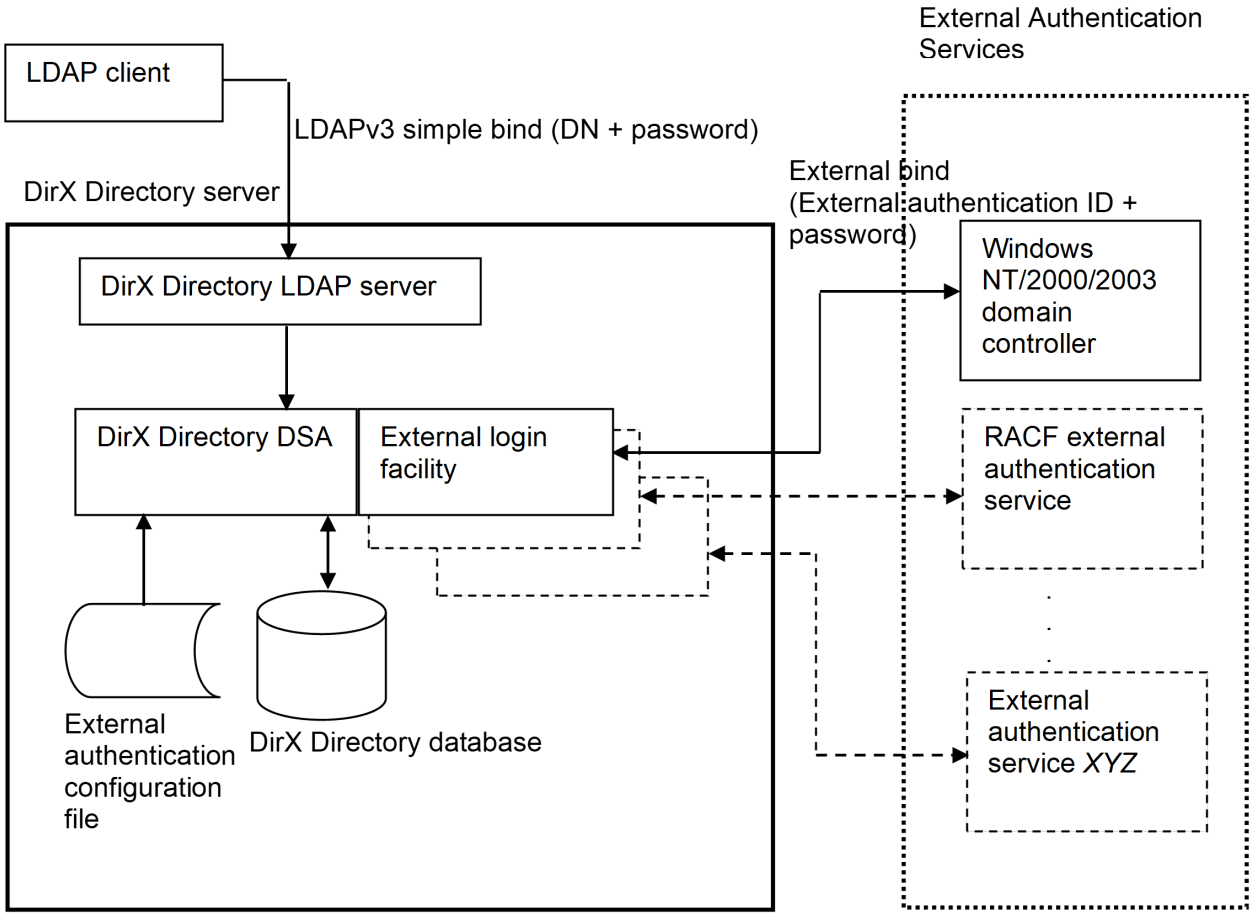



*Figure 37. Performing External Authentication*

As shown in the figure, the DirX Directory service checks an incoming simple bind request against local configuration information contained in an external authentication configuration file to determine if the bind request matches the pre-requisites for external authentication specified in the file and either authenticates the request locally according to the X.500 bind procedure - the DSA checks the DN and user password transmitted in the bind credentials against the DirX Directory database - or forwards the credentials to an external authentication service. The local configuration also describes mapping rules to be applied to the DN prior to forwarding it to the external service and/or rules to be applied to

map the externally authenticated DN onto the X.500 access control identity after successful external authentication.

## 13.1. Why Use External Authentication?

External authentication permits users to establish authenticated connections to the directory service (and thus to have access to entries in the DIT) without the need for special set of directory service credentials.The target is not to provide a single-sign-on service.

## 13.2. What Kinds of Bind Requests Can Be Externally Authenticated?

In general, only simple LDAPv3 bind operations targeted to the DirX Directory LDAP server and performed over LDAP or SSL/TLS-protected LDAP can be forwarded for external authentication.Bind requests performed using another credential choice – like the simple protected bind operations performed directly over DAP by the **dirxcp** command line tool – are not subject to forwarding, nor are bind operations performed via RPC by the administrative command line tool **dirxadm**.

## 13.3. How Is External Authentication Controlled?

The information that controls how external authentication is performed is stored in the external authentication configuration file *install_path*/**server/conf/dirxextauth.cfg**.The DirX Directory DSA looks for this file on startup and follows the configuration information specified in the file.You must create this file to enable external authentication.

The external authentication configuration file contains information about:

- The external authentication service to use
- The LDAP servers that are allowed to forward simple bind requests for external authentication
- The procedure to use if an external authentication fails
- The conditions that must be met in order to forward a particular bind and the rules for constructing the target system's account name.

The *DirX Directory External Authentication* provides detailed usage information about the external authentication configuration file and its parameters and provides some examples.

## 13.4. What External Authentication Services are Supported?

The DirX Directory service currently supports the following external authentication services:

- Windows authentication, where the DirX Directory service runs on a Windows platform and the DirX Directory DSA process checks incoming bind requests and conditionally forwards them to the Windows domain controller by calling the Windows LogonUser

API.

- RACF authentication, where the DirX Directory service is integrated with the RACF service via LDAPv3 access to the Secure-Way LDAP server running on a RACF host.

Note that the DirX Directory service can handle only one external authentication service at a time.

### 13.4.1. How Does External Authentication to Windows Work?

When you configure for Windows external authentication, you specify the OID of an attribute type that holds an entry's Windows account information - the domain name and the username – in a parameter in the configuration file (named WIN_ACCOUNT_AT). This attribute type must be indexed and its syntax must be Directory String.

The DirX Directory DSA supports two strategies for determining whether a particular bind request is subject to forwarding to Windows:

- Credentials-DN-Strategy - the bind credentials contain the Windows user and domain name and the mapping onto an X.500 entry DN is performed after the external authentication succeeds. If the configuration enables this strategy, each simple bind request that fulfills the following condition is forwarded to Windows: the DN in the credentials consists of one RDN and the type of this RDN is the WIN_ACCOUNT_AT attribute type.

  The DN supplied in the bind credentials is converted into a Windows user name and domain name. If Windows is able to authenticate that user with the user password provided in the bind credentials, the DSA maps the Windows user/domain name onto a X.500 access control identity: the DSA searches its local DIT to retrieve the entry that holds the specified WIN_ACCOUNT_AT value. If this search returns exactly one entry, its DN is the X.500 access control identity, otherwise the external bind procedure fails.

- Attribute-Entry-Strategy - the bind credentials contain the X.500 entry DN and the mapping onto the Windows user and domain name is performed based on the value of the entry's WIN_ACCOUNT_AT attribute before the external authentication function is called. If the configuration enables this strategy, each simple bind request causes the DSA to read the WIN_ACCOUNT_AT attribute of the entry in the local DIT that is named by the DN specified in the bind credentials. If the entry stores exactly one value in the Windows account attribute, this value is converted into a Windows username and domain name. The retrieved data, username and domain name and the bind password are delivered to the Windows authentication service. On success, the entry named by the original bind DN is considered to be the X.500 access control identity.

The Windows authentication service runs on a Windows domain controller. If the DSA is expected to authenticate users who are registered in domains other than the one in which the DSA is located, an inter-domain trust relationship must be established between the affected Windows domains.

### 13.4.2. How Does RACF Authentication Work?

The DSA decides whether to forward a particular simple bind request based on

configurable conditions that concern:

- The DN contained in the incoming bind credentials
- The attribute information stored in the entry specified by the DN contained in the incoming bind credentials

A set of mapping parameters in the external authentication configuration file controls the conversion from the incoming DN to the account name recognized by the external RACF system.

The DirX Directory DSA supports

- either dynamic DN mapping which involves DN resolution performed by the remote RACF LDAP server based on RDN values from the incoming bind operation
- or local DN mapping based on attributes stored at the entry in the DIT of the local DSA.

The bind operation may also invoke a change-password operation on the RACF system depending on the content of the user password in the bind credentials.

# 14. Managing Miscellaneous Scenarios

This chapter provides information on how to set up and manage a number of specialized scenarios, including how to:

- Set up and manage nested groups
- Set up and manage dynamic groups
- Manage huge search results with size limits and simple paging
- Set up NAT support in presentation addresses
- Set up IPv6 support in presentation addresses
- Set up encrypted X.500 protocols (IDMS support)
- Change the host IP address of a DirX Directory server in an existing DirX Directory installation
- Set up and manage time-based one-time password two-factor authentication (TOTP 2FA) for DirX Directory users

## 14.1. Working with Nested Groups

This section describes how to set up and maintain nested groups, which are hierarchies of imported groups.To create a nested group import hierarchy, you need to add the following items to your group definitions:

- The dirxImportGroup auxiliary object class, which extends the group to support an imported group hierarchy
- The dirxImportFrom attribute, which specifies the DN(s) of the groups to be imported

The DirX Directory DSA interprets the members of the imported groups as members of the importing group; as a result, they are also evaluated by the DirX Directory DSA's access control decision operation and its user policy management function.

Note that nested groups can only be created for entries with the groupOfNames object class.

The following figure illustrates a nested group:

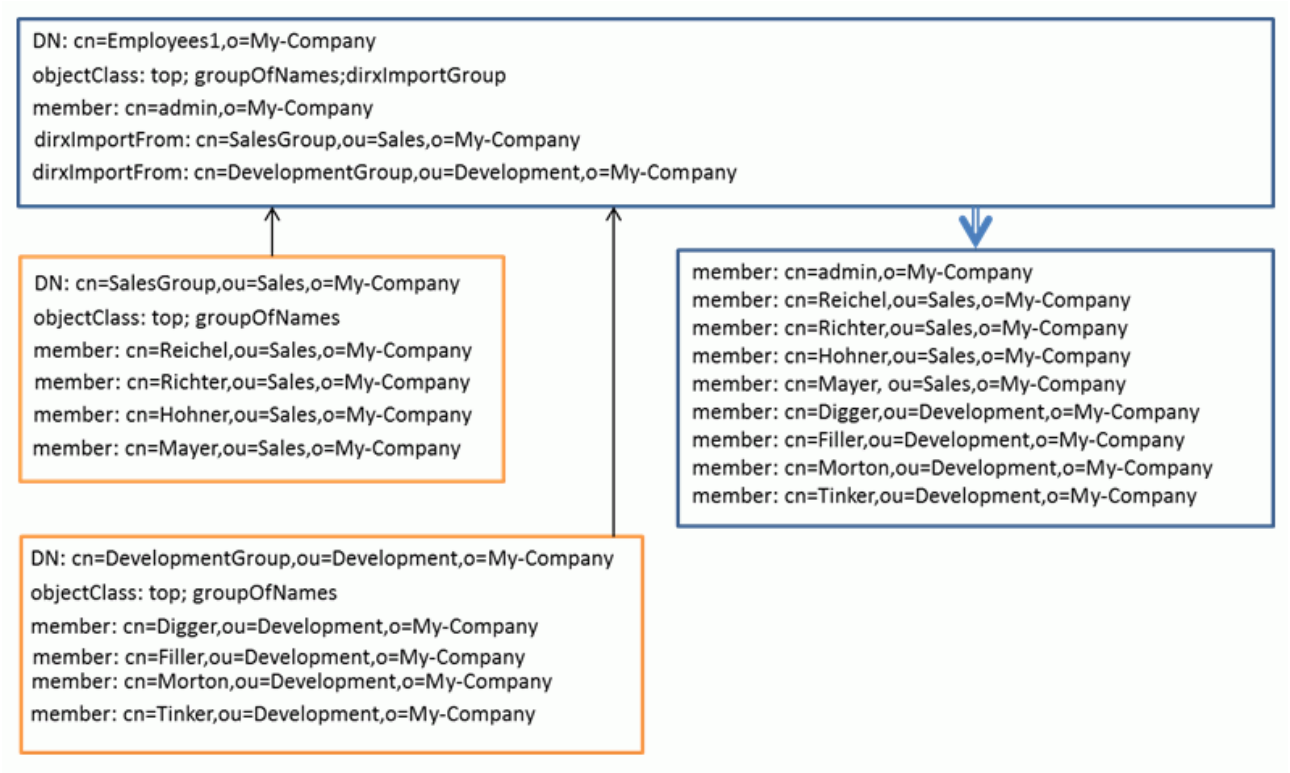



*Figure 38. Nested Groups: an Example*

As shown in the figure, the nested group **cn=Employees1,o=My-Company** imports the groupOfNames entries **cn=SalesGroup** and **cn=DevelopmentGroup**. Importing the two groups has the same effect for the nested group **cn=Employees1,o=My-Company** as it would have if this group actually contained the member attributes of all three groups.

The DirX Directory DSA's access control decision and user management functions calculate the membership of specific DNs in nested groups along with the direct memberships. For example, suppose the user with the DN **cn=reichel,ou=sales,o=My-Company** is bound to the DSA. When the DSA's access control decision operation verifies the membership of this user, it determines that the user **cn=reichel** is a member of the group **cn=SalesGroup** and a member of the group **cn=Employees1**, because **cn=Employees1** imports the members of the group **cn=SalesGroup**. The user policy management function calculates the same membership for the user **cn=reichel** as the access control decision operation does.

### 14.1.1. About the Nested Groups Object Class

The dirxImportGroup auxiliary object class extends a groupOfNames entry with the dirxImportFrom attribute to import member values by referencing other groupOfNames entries.

### 14.1.2. About the Nested Groups Attribute Types

The following attribute types are part of DirX Directory support for nested groups:

- dirxImportFrom
- dirxMember
- dirxMemberOf

- dirxImportedGroups
- dirxExportTo

The next sections provide more information about these attribute types.

#### 14.1.2.1. About the dirxImportFrom Attribute Type

The dirxImportFrom attribute specifies the DN(s) of a groupOfNames entry from which member attribute values are to be imported. To provide for complete nested group support, the imported group should be hosted on the same physical DSA as the importing group. If this is not the case, the DirX Directory DSA will ignore the imported group at evaluation time.

The dirxImportFrom attribute is a User Applications attribute and is maintained by a DirX Directory administrator. Since the imported groups themselves can in turn import other groups, the DirX Directory DSA interprets and evaluates the groups recursively through multiple levels. In a nested group import hierarchy, the maximum import hierarchy level is limited to 8 and the creation of loops is prohibited. If a cyclical reference is detected in a create- or modify-entry operation, the DSA rejects the operation with an “unwilling to perform” error and the LDAP message “[DSA]: #Loop in nested group import hierarchy detected#”.

#### 14.1.2.2. About the dirxMember Attribute Type

The dirxMember attribute is a directory operational attribute. Its values are calculated at runtime. The dirXMember attribute is synonymous with the member attribute but is a required attribute for showing all member values of a nested group in one search operation. To facilitate migration from existing DirX Directory installations, the dirxMember attribute has the same access rights as the member attribute. Requesting the dirxMember attribute in a search operation shows all values from the member attribute of a groupOfNames entry plus all values which come from the imported groups referenced recursively by the dirxImportFrom attribute. The dirxMember attribute can’t be specified in update operations. To add or delete a value, you must specify the member attribute. You can specify the dirxMember attribute in a filter expression. The search result returns the groupOfNames entries where the dirxMember attribute virtually matches.

#### 14.1.2.3. About the dirxMemberOf Attribute Type

The dirxMemberOf attribute is a directory operational attribute. Its values are calculated at runtime. Requesting the dirxMemberOf attribute in a search operation shows the DNs of groupOfNames entries where the resulting entry is either a direct member of this group or an indirect member by virtue of a nested group. Specifying dirxMemberOf in a filter expression returns the dirxMember of the specified group in the search result.

#### 14.1.2.4. About the dirxImportedGroups Attribute Type

The dirxImportedGroups attribute is a directory operational attribute. Its values are calculated at runtime. Requesting the dirxImportedGroups attribute in a search operation shows the list of DNs of all imported groupOfNames entries.

#### 14.1.2.5. About the dirxExportTo Attribute Type

The dirxExportTo attribute specifies the DN of a groupOfNames entry to maintain a link from the imported group to the importing group. The dirxExportTo attribute is a directory operational attribute and is maintained by the DSA.

### 14.1.3. Configuring Permissions for Nested Group Attributes

To use the dirxMemberOf and dirxImportedGroups directory operational attributes, you need to specify explicit permissions in the access control policies; this step is not required for the dirxMember attribute. Since the DirX Directory DSA maintains the dirxExportTo directory operational attribute automatically, this step is also not required for this attribute.

### 14.1.4. Configuring the INITIAL Index for the Member Attribute

To implement complete nested group support, you must configure the INITIAL index for the member attribute; otherwise, search operations may return with error, return incorrect or empty results or last a long time. See the **dirxadm db attrconfig** operation in the *DirX Directory Administration Reference* for more information.

### 14.1.5. Creating a Nested Group

One way to build a nested group is to create a new one with **dirxcp create**. For example, suppose the following three groups exist as groupOfNames entries:

- **My-Company** → **Sales** → **SalesGroup**
- **My-Company** → **Development** → **DevelopmentGroup**
- **My-Company** → **AdministratorGroup**

A **dirxcp search** operation on each group shows the following members:

```
dirxcp> search cn=SalesGroup,ou=Sales,o=My-Company -base -attr member
-p
```

The sample output is as follows:

```
1) cn=SalesGroup,ou=Sales,o=My-Company
    member : cn=Reichel,ou=Sales,o=My-Company
           : cn=Richter,ou=Sales,o=My-Company
           : cn=Hohner,ou=Sales,o=My-Company
           : cn=Mayer,ou=Sales,o=My-Company
```

```
dirxcp> search cn=DevelopmentGroup,ou=Development,o=My-Company -base
```

```
-attr \ member -p
```

The sample output is as follows:

```
1) cn=DevelopmentGroup,ou=Development,o=My-Company
    member : cn=Digger,ou=Development,o=My-Company
           : cn=Filler,ou=Development,o=My-Company
           : cn=Morton,ou=Development,o=My-Company
           : cn=Tinker,ou=Development,o=My-Company
```

```
dirxcp> search cn=AdministratorGroup,o=My-Company -base -attr member
-p
```

The sample output is as follows:

```
1) cn= AdministratorGroup,o=My-Company
    member : cn=admin,o=My-Company
```

The following **dirxcp create** operation creates a new nested groupOfNames entry **cn=Employees1,o=My-Company** that imports the **SalesGroup** and **DevelopmentGroup** groups:

```
dirxcp> create cn=Employees1,o=My-Company \
-attr {objectclass=groupOfNames;dirxImportGroup} \
      {member=cn=non-existent-user,o=My-Company} \
      {dirxImportFrom=cn=SalesGroup,ou=Sales,o=My-Company;
       cn=DevelopmentGroup,ou=Development,o=My-Company}
```

Since the member attribute is a mandatory attribute for a group definition, a value for it must be specified. In our example, the DN **cn=non-existent-user,o=My-Company** is specified as a placeholder and an entry with this DN doesn't exist.

Another way to build a nested group is to modify an existing groupOfNames entry by adding the dirxImportGroup auxiliary object class to it and optionally specifying one or more values for the dirxImportFrom attribute. (Since the dirxImportFrom attribute is an optional attribute, this step can be omitted.)

For example, the following **dirxcp modify** operation extends the existing group **cn=AdministratorGroup,o=My-Company** to support an imported group hierarchy and imports the group **cn=DevelopmentGroup,ou=Development,o=My-Company**:

```
dirxcp> modify cn=AdministratorGroup,o=My-Company \
-add {objectclass=dirxImportGroup} \
     {dirxImportFrom=cn=DevelopmentGroup,ou=Development,o=My-Company}
```

### 14.1.6. Listing All Members of a Nested Group

To view the members of a nested group, including all the members from the imported groups, specify the dirxMember attribute in the **dirxcp** search request. For example, to show all the members in the nested group **cn=Employees1,o=My-Company**, run the following **dirxcp search** operation:

```
dirxcp> search cn=Employees1,o=My-Company -base -attr dirxMember -p
```

The sample output is as follows:

```
1) cn=Employees1,o=My-Company
    dirxMember : cn=non-existent-user,o=My-Company
               : cn=Reichel,ou=Sales,o=My-Company
               : cn=Richter,ou=Sales,o=My-Company
               : cn=Digger,ou=Development,o=My-Company
               : cn=Filler,ou=Development,o=My-Company
               : cn=Morton,ou=Development,o=My-Company
               : cn=Tinker,ou=Development,o=My-Company
               : cn=Hohner,ou=Sales,o=My-Company
               : cn=Mayer,ou=Sales,o=My-Company
```

Specifying the dirxMember attribute also works for “normal” groups. For example:

```
dirxcp> search cn=SalesGroup,ou=Sales,o=My-Company -base -attr
dirxmember -p
```

The sample output is as follows:

```
1) cn=SalesGroup,ou=Sales,o=My-Company
    dirxMember : cn=Reichel,ou=Sales,o=My-Company
               : cn=Richter,ou=Sales,o=My-Company
               : cn=Hohner,ou=Sales,o=My-Company
               : cn=Mayer,ou=Sales,o=My-Company
```

### 14.1.7. Searching for Group Membership in Groups and Nested Groups

To search for an entry's membership in groups and nested groups, specify the dirxMember attribute in the **dirxcp search** filter. For example:

```
dirxcp> search o=My-Company -sub -fi
(dirxmember=cn=Richter,ou=Sales,o=My-Company) -p
```

The sample output is as follows:

```
 1) cn=SalesGroup,ou=Sales,o=My-Company
 2) cn=Employees1,o=My-Company
```

### 14.1.8. Listing User Membership in Groups and Nested Groups

To list the groups and nested groups to which an entry belongs, specify the dirxMemberOf attribute in the **dirxcp search** request. For example:

```
dirxcp> search cn=Reichel,ou=Sales,o=My-Company -base -attr
dirxMemberOf -p
```

The sample output is as follows:

```
1) cn=Reichel,ou=Sales,o=My-Company
    dirxMemberOf : cn=SalesGroup,ou=Sales,o=My-Company
                 : cn=Employees1,o=My-Company
```

### 14.1.9. Determining Specific Entry-Nested Group Membership

There are several ways to determine the membership of a specific entry in a specific group or specific nested group. One method is to perform a compare operation. For example:

```
dirxcp> compare cn=Employees1,O=My-Company -attr \
 {dirxmember=cn=reichel,ou=sales,o=my-company}
```

The sample output is as follows:

```
M=TRUE
```

The result **M=TRUE** indicates that the DN **cn=reichel,ou=sales,o=my-company** is a member of the nested group **cn=Employees1,O=My-Company**.

The compare operation also works for “normal” groups. For example:

```
dirxcp> compare cn=SalesGroup,ou=Sales,o=My-Company -attr \
{dirxmember=cn=reichel,ou=sales,o=my-company}
```

The sample output is as follows:

```
M=TRUE
```

The result **M=TRUE** indicates that the DN **cn=reichel,ou=sales,o=my-company** is a member of the group **cn=SalesGroup,ou=Sales,o=My-Company**.

Another way is to perform a base-level search and specify the dirxMember attribute in the filter. For example:

```
dirxcp> search cn=Employees1,O=My-Company -base -fi \
 {dirxmember=cn=reichel,ou=sales,o=my-company}
```

The sample output is as follows:

```
1) cn=Employees1,o=My-Company
```

If the DN of the group is returned, the filter has matched and indicates that the entry is a member of the nested group.

### 14.1.10. Searching for a Group’s DirxMember

To search for the dirxMember of a group, specify the dirxMemberOf attribute in the **dirxcp search** filter. For example:

```
dirxcp> search o=My-Company -sub -fi (dirxMemberOf=
cn=Employees1,o=My-Company) -p
```

The sample output is as follows:

```
1)cn=Reichel,ou=Sales,o=My-Company
2)cn=Richter,ou=Sales,o=My-Company
3)cn=Digger,ou=Development,o=My-Company
```

```
4)cn=Filler,ou=Development,o=My-Company
5)cn=Morton,ou=Development,o=My-Company
6)cn=Tinker,ou=Development,o=My-Company
7)cn=Hohner,ou=Sales,o=My-Company
8)cn=Mayer,ou=Sales,o=My-Company
```

As you can see in this search example, the member with the DN **cn=non-existing-user,o=My-Company** is not part of the result, because a search result shows only the DNs of entries which really exist.

### 14.1.11. Determining a Nested Group's Imported Groups

To return the complete import hierarchy of a nested group, specify the dirxImportedGroups attribute in the **dirxcp search** operation. For example:

```
dirxcp> search cn=Employees1,o=My-Company -base -attr
dirxImportedGroups -p
```

The sample output is as follows:

```
1) cn=Employees1,o=My-Company
    dirxImportedGroups : 1- cn=Employees1,o=My-Company
                       : 2-- cn=SalesGroup,ou=Sales,o=My-Company
                       : 2-- cn=DevelopmentGroup,ou=Development,o=My-
Company
```

### 14.1.12. Adding a Member to a Nested Group

To add a member to a nested group, either add a new value to the local member attribute or add a new value to the member attribute in one of the imported groups. No matter which method you choose, be aware that adding a new member value may change the membership of other nested groups.

The following **dirxcp modify** operation adds a new value to the member attribute locally in the nested group **cn=Employees1**:

```
dirxcp> modify cn=Employees1,o=My-Company \
-add {member=cn=Abele,ou=Sales,o=My-Company}
```

The following **dirxcp modify** operation adds a new value to the member attribute in the nested group **cn=SalesGroup**:

```
dirxcp> modify cn=SalesGroup,ou=Sales,o=My-Company \
-add {member=cn=Abele,ou=Sales,o=My-Company}
```

Independent of the method used, the nested group **cn=Employees1** now incorporates the new member:

```
dirxcp> search cn=Employees1,o=My-Company -base -attr dirxMember -p
```

The sample output is as follows:

```
1) cn=Employees1,o=My-Company
    dirxMember : cn=non-existent-user,o=My-Company
               : cn=Reichel,ou=Sales,o=My-Company
               : cn=Richter,ou=Sales,o=My-Company
               : cn=Digger,ou=Development,o=My-Company
               : cn=Filler,ou=Development,o=My-Company
               : cn=Morton,ou=Development,o=My-Company
               : cn=Tinker,ou=Development,o=My-Company
               : cn=Hohner,ou=Sales,o=My-Company
               : cn=Mayer,ou=Sales,o=My-Company
               : cn=Abele,ou=Sales,o=My-Company
```

### 14.1.13. Removing a Member from a Nested Group

Removing a member from a nested group can be challenging in a complex nested group configuration because you may need to remove a member from multiple imported groups to delete the membership in the nested group.On the other hand, deletion of a member may also change the membership of other nested groups.

In this example, the member **cn=Mayer,ou=Sales,o=My-Company** is removed from the nested group **cn=Employees1** by simply removing **cn=Mayer,ou=Sales,o=My-Company** as a member of the group **cn=SalesGroup**.For example:

```
dirxcp> modify cn=SalesGroup,ou=Sales,o=My-Company \
-rem {member=cn=Mayer,ou=Sales,o=My-Company}
```

## 14.2. Working with Dynamic Groups

This section describes how to set up and maintain dynamic groups.A dynamic group is defined as an entry with the structural object class dirxGroupOfURLs and zero or more

dirxMemberUrl attribute values.Each dirxMemberUrl value describes a set of entries in the local DSA by its search expression.The set union of these entries build the members of the dynamic group.An entry is considered to be a member of the dynamic group if it falls within the search scope and matches the search filter of at least one dirxMemberUrl value.Only entries with DSE-Type ENTRY can be member of a dynamic group.

The DirX Directory DSA's access control decision function and the user policy management function support dynamic groups.That is the DN of a dynamic group can be specified in the user group component of an ACI-Item respectively username component of a user policy and will be recognized as a dynamic group.At run time the attributes of the evaluated entry respectively user are evaluated against the dirxMemberUrl values of the dynamic group.Make sure that the attributes applied in the search filter cannot be modified by the users themselves.Otherwise, they could inadvertently become member of a dynamic group by modifying the attributes in their entries.

### 14.2.1. About the dirxGroupOfURLs Object Class

The dirxGroupOfURLs structural object class defines a dynamic group.A combination with the auxiliary nested group object class dirxImportGroup is not supported.

### 14.2.2. About the dirxMemberUrl Attribute

The dirxGroupOfURLs object class uses the multi valued attribute dirxMemberUrl to define the LDAP search expression. The members are specified by a base DN, the scope of the search and a search filter. There is no need to maintain a list of members manually. Group membership is calculated by the definition of LDAP search expression each time when group membership is evaluated. See the *DirX Directory Syntaxes and Attributes* for more details about the dirxMemberUrl attribute and its syntax. Also some restrictions may apply to the used attribute types and filter expressions. Please refer to the actual DirX Directory Release Notes of your deployed DirX Directory server.

### 14.2.3. Configuring Attribute Indices for the Attributes in the Search Filter

For searching and listing the members of a dynamic group it is highly recommended to perform appropriate index configuration for the attributes specified in the search filter. See the **dirxadm db attrconfig** operation in the *DirX Directory Administration Reference* for more information on how to optimize search filter operations with attribute indices.

### 14.2.4. Creating a Dynamic Group

The following **dirxcp create** operation creates a new dynamic group. The dirxMemberUrl value specifies **o=My-Company** as the search base, value **sub** as the search scope which will incorporate all entries below the search base and the search filter **(&(l=sunnyvale)(title=senior*))**.

```
dirxcp> create cn=DG1,o=My-Company
{objectClass=dirxGroupOfUrls}
```

```
{dirxMemberUrl=ldap:///o=My-
Company??sub?(&(l=sunnyvale)(title=senior*))}
```

### 14.2.5. Listing all Members of a Dynamic Group

To display all members of a dynamic group, request the attribute dirxMember in the **dirxcp show** or **search** operation. For example to display all members of the dynamic group **cn=DG1,o=My-Company**, run the following search operation:

```
dirxcp> search cn=DG1,o=My-Company -base -attr dirxMember -pretty
```

The sample output is as follows:

```
1) cn=DG1,o=My-Company
    dirxMember : cn=Lilin Markham,ou=Accounting,o=My-Company
               : cn=Winston Yoakum,ou=Accounting,o=My-Company
               : cn=Arluene Management,ou=Accounting,o=My-Company
               : cn=Shandeigh Combaz,ou=Accounting,o=My-Company
               : cn=Ayn Skedelsky,ou=Product Development,o=My-Company
               : cn=Andrew Herring,ou=Human Resources,o=My-Company
               : cn=Les Dagg,ou=Payroll,o=My-Company
               : cn=Alfreda Swanston,ou=Payroll,o=My-Company
```

Please take into account that all members will be returned in one result and this fact may lead into memory shortage situations in the DirX Directory DSA and in the DirX Directory LDAP server if the list of dirxMember values becomes very large.

### 14.2.6. Searching for Members of a Dynamic Group

Another way to retrieve members of a dynamic group is to specify the attribute dirxMemberOf in the search filter. This method has the advantage that it can be combined with the simple paging mechanism and depending on the page size the result is delivered in multiple pages and thus reduces overall peak memory consumption.

The following example shows how to search for the members of dynamic group **cn=DG1,o=My-Company** by specifying attribute dirxMemberOf in the search filter.

```
dirxcp> search / -sub -fi {dirxMemberOf=cn=DG1,o=My-Company} -p
```

The sample output is as follows:

```
 1) cn=Lilin Markham,ou=Accounting,o=My-Company
```

```
 2) cn=Winston Yoakum,ou=Accounting,o=My-Company
 3) cn=Arluene Management,ou=Accounting,o=My-Company
 4) cn=Shandeigh Combaz,ou=Accounting,o=My-Company
 5) cn=Ayn Skedelsky,ou=Product Development,o=My-Company
 6) cn=Andrew Herring,ou=Human Resources,o=My-Company
 7) cn=Les Dagg,ou=Payroll,o=My-Company
 8) cn=Alfreda Swanston,ou=Payroll,o=My-Company
```

The next example shows how to search for the members of dynamic group **cn=DG1,o=My-Company** by specifying attribute dirxMemberOf in the search filter in combination with simple paging.

```
dirxcp> search / -sub -fi {dirxMemberOf=cn=DG1,o=My-Company} -p
-vtype simple -vpagesize 4
```

The sample output displays the first page of the result:

```
 1) cn=Lilin Markham,ou=Accounting,o=My-Company
 2) cn=Winston Yoakum,ou=Accounting,o=My-Company
 3) cn=Arluene Management,ou=Accounting,o=My-Company
 4) cn=Shandeigh Combaz,ou=Accounting,o=My-Company
Incomplete operation:
    Partial-Outcome-Qualifier
      Query-Reference : \0x00\0x00\0x00\0x10
```

To display the next page of the result perform a **nextpage** operation:

```
dirxcp> nextpage -vpagesize 4 -p
```

**dirxcp** displays the next page of the sample output:

```
 1) cn=Ayn Skedelsky,ou=Product Development,o=My-Company
 2) cn=Andrew Herring,ou=Human Resources,o=My-Company
 3) cn=Les Dagg,ou=Payroll,o=My-Company
 4) cn=Alfreda Swanston,ou=Payroll,o=My-Company
```

### 14.2.7. Searching for Group Membership in Dynamic Groups

To search for an entry's membership in dynamic groups specify the dirxMember attribute

in the **dirxcp** search filter. For example:

```
dirxcp> search / -sub -fi {dirxMember=cn=Winston
Yoakum,ou=Accounting,o=My-Company} -p
```

The sample output is as follows:

```
1) cn=DG1,o=My-Company
```

### 14.2.8. Listing User Membership in Dynamic Groups

To list the dynamic groups to which an entry belongs, explicitly request attribute dirxMemberOf in the **dirxcp search** or **show** request.

In the following example requesting the attribute dirxMemberOf shows in which dynamic group(s) the entry **cn=Winston Yoakum,ou=Accounting,o=My-Company** is member of:

```
dirxcp> search {cn=Winston Yoakum,ou=Accounting,o=My-Company} -base
-attr dirxmemberof -p
```

In the sample output you can see that Winston Yoakum belongs to the dynamic group **cn=DG1,o=My-Company**:

```
1) cn=Winston Yoakum,ou=Accounting,o=My-Company
    dirxMemberOf : cn=DG1,o=My-Company
```

### 14.2.9. Determining Specific Entry Dynamic Group Membership

There are several ways to determine the membership of a specific entry in a specific dynamic group. One method is to perform a **dirxcp compare** operation.

In the following example the result **M=TRUE** indicates that the **DN cn=Winston Yoakum,ou=Accounting,o=My-Company** is a member of the dynamic group **cn=DG1,o=My-Company**:

```
dirxcp> compare cn=DG1,o=My-Company -attr {dirxMember=cn=Winston
Yoakum,ou=Accounting,o=My-Company}
```

The sample output is as follows:

```
M= TRUE
```

The **compare** operation can be arranged in another way. For example:

```
dirxcp> compare {cn=Winston Yoakum,ou=Accounting,o=My-Company} -attr
{dirxMemberOf=cn=DG1,o=My-Company}
M= TRUE
```

An alternative way is to perform a base level search against the dynamic group and specify the dirxMember attribute in the filter. For example:

```
dirxcp> search cn=DG1,o=My-Company -base -fi {dirxMember=cn=Winston
Yoakum, ou=Accounting, o=My-Company} -p
```

The sample output displays the name of the dynamic group **cn=DG1,o=My-Company** that Winston Yoakum belongs to:

```
1) cn=DG1,o=My-Company
```

If the DN of the group is returned, the filter matched and indicates that the entry is a member of the dynamic group as shown in the example above.

Another way to evaluate the group membership is to perform a base level search with attribute dirxMemberOf in the filter. For example:

```
dirxcp> search {cn=Winston Yoakum,ou=Accounting,o=My-Company} -base
-fi {dirxMemberOf=cn=DG1,o=My-Company} -p
```

The sample output displays the name of member Winston Yoakum:

```
1) cn=Winston Yoakum, ou=Accounting, o=My-Company
```

If the DN of the entry is returned, the dirxMemberOf filter matched and indicates that the entry is a member of the dynamic group as shown in the example above.

### 14.2.10. Performance Considerations

This section discusses the performance of several operations for determining dynamic group membership.

#### 14.2.10.1. Listing versus Searching Members of a Dynamic Group

Requesting the attribute dirxMember in a **search** operation from a dynamic group will internally perform a search operation using the filter from the dirxMemberUrl attribute; for example **search** *DN-of-Dynamicgroup* **-base -attr dirxMember**. The duration of the search, the CPU load and the main memory utilization depends on the number of dirxMemberUrl attribute values (each value launches a search operation), the size of your database, the attribute index configuration and the number of resulting dirxMember values. In case of a large number of resulting dirxMember values, it would be better to perform a search operation with dirxMemberOf specified in the filter and simple paging; for example *search / -sub -fi dirxMemberOf=**DN-of-Dynamicgroup* **-vtype simple -vpagesize** *Page-Size*. This operation will not perform faster, but it will be easier - due to less memory consumption - to handle a very large number of members of the dynamic group.

Anyway avoid to request attribute dirxMember where multiple dynamic groups will show up in the search result; for example **search / -sub -fi (objectclass=dirxGroupOfURLs) -attr dirxMember**.

In case it is intended to obtain the list of dirxMember values for the purpose of dynamic group membership evaluation for a specific entry, then the compare operation is the better solution.

#### 14.2.10.2. Searching for Group Membership in Dynamic Groups

The duration of a search request with the attribute dirxMember in the search filter increases with the number of dynamic groups; for example **search / -sub -fi (dirxMember=cn=admin,o=My-Company)**. That is, for every known dynamic group the entry specified in the dirxMember assertion is checked explicitly for membership match against the dynamic group. The more dynamic groups are present the longer the search will last. If it is intended to list the number of dynamic groups an entry belongs to, then start a base level search and request attribute dirxMemberOf. Typically, this will perform better than search with dirxMember in the filter.

#### 14.2.10.3. Listing User Membership in Dynamic Groups

Requesting attribute dirxMemberOf shows the membership in dynamic groups of the entry; for example **search cn=admin,o=My-Company -base -attr dirxMemberOf**. The duration of this operation will increase with the number of existing dynamic groups. That is, for every known dynamic group, membership is evaluated against every entry in the result. For base level searches the duration for this operation might be still acceptable. In tree searches with a large amount of resulting entries requesting attribute dirxMemberOf is not recommended.

#### 14.2.10.4. Determining Specific Entry Dynamic Group Membership

All the operations to determine the membership of a specific entry in a specific dynamic group perform very well because they evaluate only one dynamic group and one entry. These operations are:

- Compare dynamic group with attribute dirxMember; for example

```
compare cn=DG1,o=My-Company -attr {dirxMember=cn=Winston Yoakum,
ou=Accounting, o=My-Company}
```

- Compare entry with attribute dirxMemberOf; for example

```
compare {cn=Winston Yoakum, ou=Accounting, o=My-Company} -attr
{dirxMember=cn=DG1, o=My-Company}
```

- Base level search of dynamic group with attribute dirxMember in the filter; for example

```
search {cn=DG1, o=My-Company} -base -fi {dirxMember=cn=Winston
Yoakum, ou=Accounting, o=My-Company}
```

- Base level search of an entry with the attribute dirxMemberOf in the filter; for example

```
search { cn=Winston Yoakum, ou=Accounting, o=My-Company } -base
-fi {dirxMemberOf=cn=DG1, o=My-Company}
```

### 14.2.11. Dynamic Groups versus Static Groups

Static groups using the structural object class groupOfUniqueNames are not highly optimized.Determining group membership and modifications to the uniqueMember attribute increases with the number of values.As a result, we recommend avoiding this kind of static group if possible.In the following discussion and comparison, groupOfUniqueNames will not be considered.

Static groups using the object class groupOfNames are highly optimized.Even with millions of member attribute values, the time for determining group membership and updating values remains constantly low.Pure static groups perform best in determining group membership of an entry.The only drawback is bulk updates with thousands of value changes in one operation in combination with a member attribute index configuration.In the worst case, this operation can lead to main memory or DBAM cache bottlenecks.

Nested static groups using the object class groupOfNames plus the auxiliary class dirxImportGroup perform almost as well as pure static groups.There can be an advantage in updating the groups, because a smart construction of nested groups can reduce the overall number of required updates compared with a pure static group construction.But there is a slight disadvantage in the access time for determining group membership which increases with the depth of the nested group hierarchy.

In dynamic groups, membership is defined by the specification of one more LDAP search expressions consisting of a base DN, a scope and a search filter.The explicit maintenance of static lists of members is not necessary.Dynamic groups perform best in updating the

member list because there is no effort at all to maintain a static list of members.The time for determining group memberships depends on many factors, including the complexity of the search filter, the matching rules of the attributes in the filter items, and the complexity and size of the entry under evaluation.Compared with nested static groups, dynamic groups can keep up with determining group membership if the search filter is simple and can even outperform nested static groups if the depth of the nested group hierarchy is too high.

## 14.3. Managing Huge Search Results

Huge search results can exhaust DirX Directory system resources.To avoid this situation, the size limit for search operations is set to **2048** result entries by default.(See the **SL=***size_limit* component of the **User Policy** attribute in the *DirX Directory Syntaxes and Attributes* for details.)

You can also adjust the size limit of the search operation in the following ways:

- For DAP binds, by performing a **dirxcp args** operation that specifies a size limit.(See the **args (dirxcp)** reference page in the *DirX Directory Administration Reference* for details.)
- For LDAP binds, by performing a **dirxcp ldapargs** operation that specifies a size limit.(See the **ldapargs (dirxcp)** reference page in the *DirX Directory Administration Reference* for details.)
- For LDAP binds, by specifying the size limit in the LDAP Search Service Controls attribute.(See the *DirX Directory Syntaxes and Attributes* for details.)

You can manage the return of huge search results with simple paging.For paged searches, the size limit for the maximum number of result entries is **16384** by default.(See the **PRSL=***paged_result_size_limit* component of the **User Policy** attribute in the *DirX Directory Syntaxes and Attributes* for details.) The maximum number of result entries per page is **2048** by default.(See the **SPL=***single_page_limit* component of the **User Policy** attribute in the *DirX Directory Syntaxes and Attributes* for details.)

If you expect huge search results, use **dirxcp obj search** and **dirxcp obj nextpage** in the following sequence to perform a paged search:

1. Perform an initial search operation with simple paging, for example

   ```
   dirxcp> obj search o=My-Company -subtree
                         -vtype SIMPLE
                         -vpagesize 15
                         -vsortkey cn
   ```

   (See the *DirX Directory Administration Reference* for details on the options of the **obj search** operation.)

2. Perform several subsequent **obj nextpage** operations to read all pages of the search result, for example:

```
dirxcp> obj nextpage -vpagesize 15
```

(See the *DirX Directory Administration Reference* for details on the **obj nextpage** operation.)

If the last entry of a page is the partial-outcome-qualifier, there is at least one follow-up page for this search result that can be read with an **obj nextpage** operation. (See the **Partial-Outcome-Qualifier** attribute syntax in the *DirX Directory Syntaxes and Attributes* for details.)

If the last entry of a page is a real search result entry (and not the partial-outcome-qualifier), all pages of the search result have been read.

After reading the last page of a paged search result, the server discards the entire search result.

3. Alternatively, you can abort reading the pages of the search result with the command:

```
dirxcp> obj nextpage -terminate
```

The server discards the entire search result.

The server also discards the entire simple page search result when:

- The time period specified in the Paging Policy attribute has expired since the last recent **obj search** or **obj nextpage** operation.(See the **Paging Policy** attribute in the *DirX Directory Syntaxes and Attributes* for details.)
- The client performs an **obj unbind** operation.

The *DirX Directory Administration Reference* provides detailed information about the **dirxcp obj nextpage** operation and the options for the **dirxcp obj search** operation performing simple paging.

The server uses the lowest limits when performing the search operation.

# 14.4. Enabling NAT Support in PSAP Addresses

In a distributed environment, it is necessary to store the presentation (PSAP) addresses of the DSAs participating in the network; for example, in the cooperating DSA table and knowledge attributes.The DSAs use these PSAP addresses to perform shadowing (DISP) and chaining (DSP).Usually these PSAP addresses are globally valid.However, it is possible that behind a firewall, the PSAP addresses are valid only locally.When communicating with a DSA behind such a firewall, Network Address Translation (NAT) must be taken into account.

The following example illustrates such a configuration.

Recall My-Company's network configuration described in previous chapters:

- Headquarters in Munich with DSA1 (My-Company Sales and Development department) and DSA2 (shadow consumer of DSA1).(See the chapter "Creating a Shadow DSA" for details.)
- The manufacturing plant in Durach with DSA3 (My-Company Manufacturing department).(See the chapter "Distributing the DIT across Multiple DSAs" for details.)
- The acquired sales partner, the STU Company in Boston (STU Sales department).(See the chapter "Multireplication" for details.)

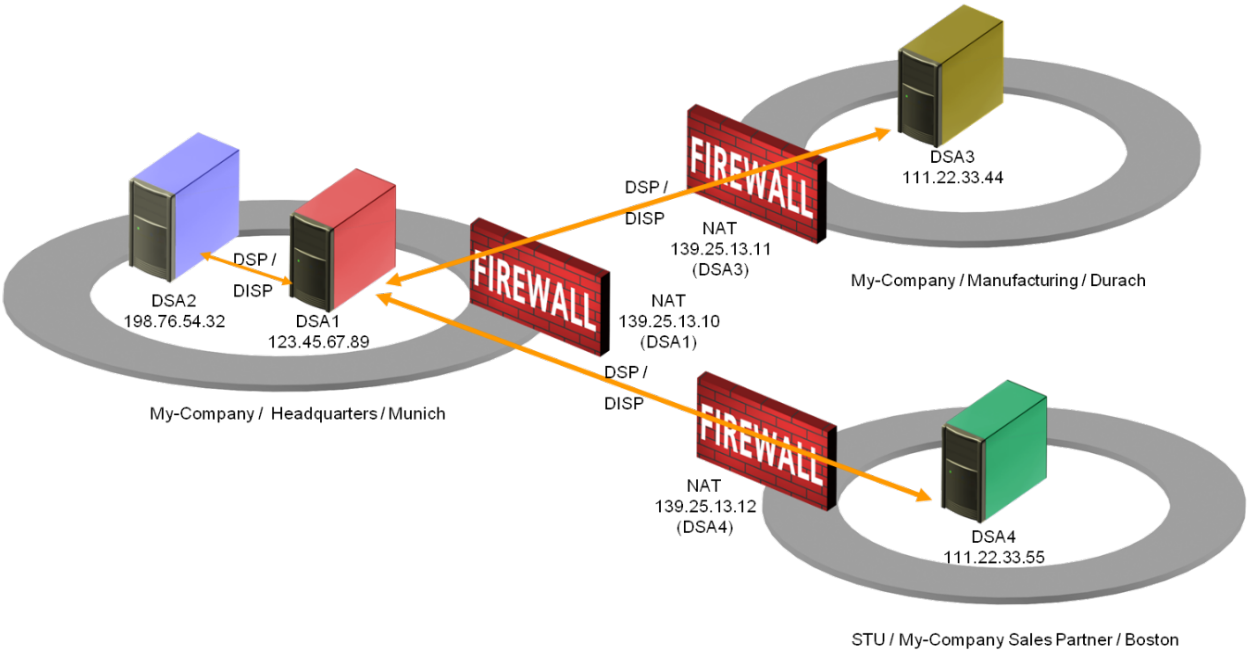



*Figure 39. Network with NAT*

As shown in the figure above, the internal networks at the locations Munich, Durach and Boston are protected by firewalls that perform network address translation. For example, DSA1 must be addressed from outside the My-Company local network in Munich using the IP address 139.25.13.10. The firewall redirects the IP packages to the address that is valid within the Munich network; that is, to 123.45.67.89. The firewall does not replace such IP addresses contained in application protocol data; for example, when transferring a referral or replicating the cooperating DSA table.

To resolve this issue, DirX Directory supports the optional use of **DNS** components in attributes with Presentation-Address attribute syntax (PSAP). If a **DNS** component is contained in a PSAP address, a DSA will use the name resolution provided by the system instead of using the network address (**NA** component) contained in the PSAP.

In My-Company's network configuration, the administrators store the DNS component names "HQMUCDSA1" for DSA1, "DURDSA3" for DSA3 and "BOSDSA4" for DSA4 together with their connection data in a name service. It is not necessary to store the DNS component name of DSA2 in the name service because DSA2 just communicates with DSA1 inside the internal network. Thus DSA3 and DSA4 cannot communicate with DSA2. When performing DISP or DSP, DirX Directory automatically reads the correct IP address from the name service. See the section "Presentation-Address" in the chapter "String

Representations for DAP Binds" in the *DirX Directory Syntaxes and Attributes* for information about how DirX Directory stores the DNS component of DSAs.

My-Company has stored the following values for the PSAP addresses in the cooperating DSA table and the knowledge references:

- DSA1:

```
{TS=DSA1,NA='TCP/IP_IDM!internet=123.45.67.89+port=19999',DNS=HQMU
CDSA1+19999}
```

- DSA2:

```
{TS=DSA2,NA='TCP/IP_IDM!internet=198.76.54.32+port=21111'}
```

- DSA3:

```
{TS=DSA3,NA='TCP/IP_IDM!internet=111.22.33.44+port=23333',DNS=DURD
SA3+23333}
```

- DSA4:

```
{TS=DSA4,NA='TCP/IP_IDM!internet=111.22.33.55+port=24444',DNS=BOSD
SA4+24444}
```

Recall from the chapter "Distributing the DIT across Multiple DSAs" that the employees' master entries of the manufacturing department are stored at DSA3 in Durach:

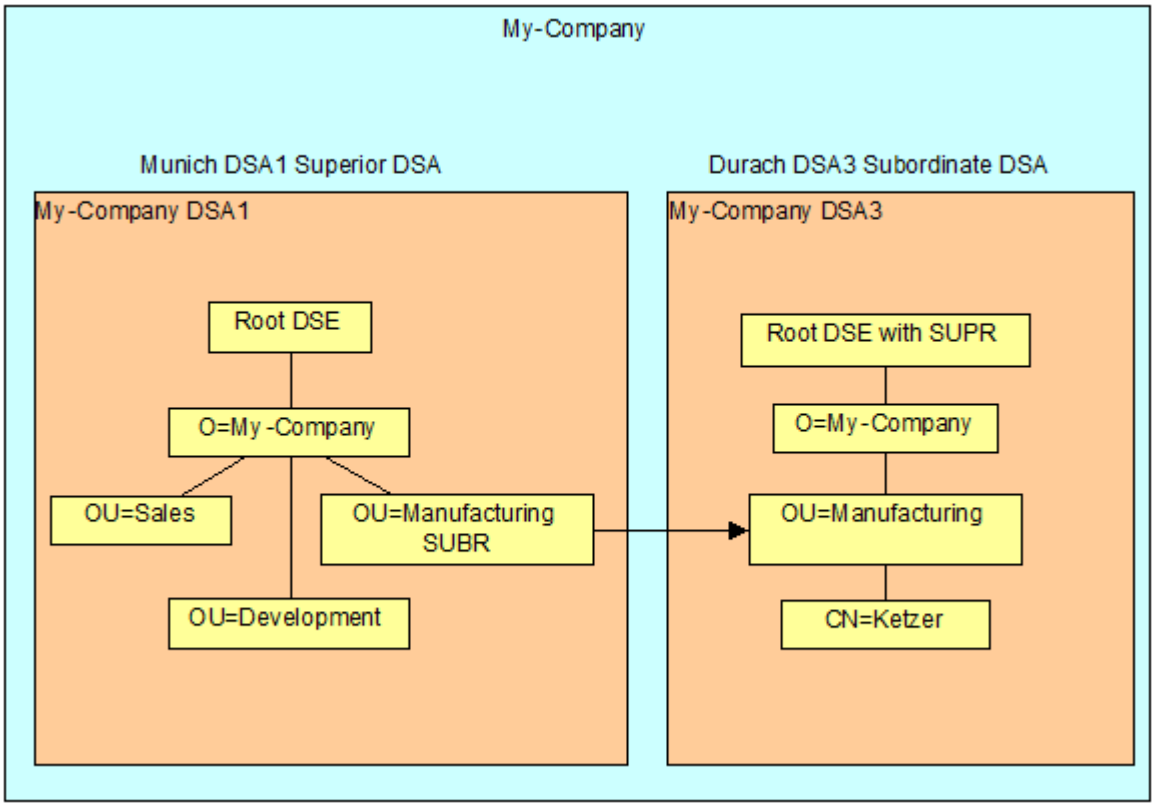



*Figure 40. My-Company Distributed DIT*

Now the administrator of DSA1 reads Mr.Ketzer's entry.DSA1 performs the following steps to read Mr.Ketzer's entry from DSA3:

1. DSA1 reads the PSAP address value of DSA3 in the specific knowledge attribute of the subordinate reference (SUBR) for the OU=Manufacturing.
2. DSA1 uses the DNS component name **DURDSA3** to request the correct IP Address from the name service.
3. DSA1 uses the port number **23333** and the IP address (**139.25.13.11**) received from the name service to read Mr.Ketzer's entry from DSA3 via DSP protocol (chaining).

# 14.5. Creating a Mixed IP Version Configuration

Starting with version V8.1, DirX Directory supports Internet Protocol version 6 (IPv6) for the IDM stack; that is, a DirX Directory DSA newer than DirX Directory V8.1 can communicate over IPv6 and Internet Protocol version 4 (IPv4) simultaneously.The following rules and restrictions apply for IPv4 and IPv6 communication:

- Only the IDM stack provides IPv6 functionality.IPv6 functionality is not available for the OSI stack.
- For IPv6 traffic, it is highly recommended to run all DSAs on dual-stack machines, which provide the IPv4 and IPv6 stacks on a single socket.This configuration ensures maximum flexibility concerning connecting different types of DSAs; for example, DSAs older than V8.1, with DSAs V8.1 or newer.
- The setting of the environment variable **DIRX_IP_STACK** specifies for which IP version of incoming traffic the DirX Directory servers listen.Make sure that you do not set this

environment variable to a value that your system does not provide.(See the chapter "Environment Variables" in the *DirX Directory Administration Reference* for details.)

- The setting of the **DNS** component of the PSAP address attribute values specifies the IP version of outgoing traffic to other DSAs.(See the section "Presentation Address" in the chapter "DirX Directory String Representation for DAP Binds" in the *DirX Directory Syntaxes and Attributes* for details.)
- You must ensure that the name service is configured correctly; that is, if two DSAs are to communicate via the IPv6 stack, the name service must return their IPv6 addresses correctly.If dual-stack communication is used, the name service must return the IPv4 and IPv6 addresses of all dual-stack machines.

  You can check to see whether the name service is configured correctly by performing the command **dirxhostinfo -I** (this is an uppercase i) [*hostname*].This command queries the name service for all IPv4 and IPv6 addresses for the *hostname* specified.If the result does not deliver an IPv6 address, this host either does not have an IPv6 address or the name service is configured incorrectly.In both cases, IPv6 communication will fail.(See the **dirxhostinfo** reference page in the *DirX Directory Administration Reference* for more details.)

The following example illustrates how to set up and configure a mixed IPv4 and IPv6 communication environment.

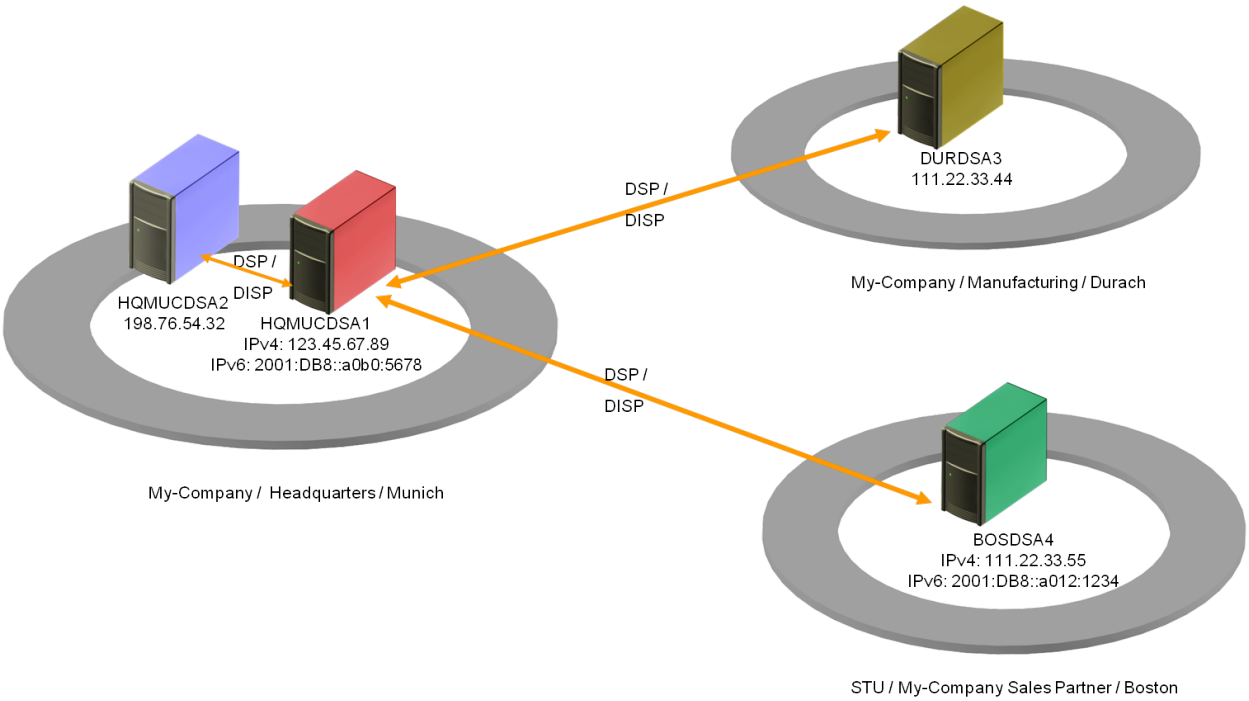
Recall My-Company's network configuration from the previous chapters:

- Headquarters in Munich with DSA1 / DNS component name: HQMUCDSA1 (My-Company Sales and Development department) and DSA2 / DNS component name: HQMUCDSA2 (shadow consumer of DSA1). (See the chapter "Creating a Shadow DSA" for details.)
- The remote manufacturing plant in Durach with DSA3 / DNS component name: DURDSA3 (My-Company Manufacturing department). (See the chapter "Distributing the DIT across Multiple DSAs" for details.)
- The acquired sales partner, the STU Company in Boston (STU Sales department). (See the chapter "Multireplication" for details.)

Headquarters in Munich wants to add the new DSA in Boston DSA4 / DNS component name: BOSDSA4 to its configuration. Recall from the chapter "Multireplication" that My-Company and STU Company decide to replicate My-Company's Sales and Development employee directory on the DSA in Boston, and to replicate the STU Company's employee directory on the DSA in Munich.

While the connections between DSA1 (HQMUCDSA1) in Munich and the new DSA4 (BOSDSA4) in Boston will use the IPv6 protocol, the connections between DSA1 and DSA2 in Munich and DSA3 in Durach run over IPv4 protocol. The following figure illustrates this configuration:

*Network with Mixed IPv4 and IPv6 Communication*




The administrators perform the following steps to connect the new BOSDSA4 to the existing My-Company network:

1. Upgrade HQMUCDSA1 to DirX Directory V8.1 or newer (dual-stack machine). The new BOSDSA4 must also install DirX Directory V8.1 or newer to use IPv6.
2. Set the environment variable **DIRX_IP_STACK=all** on both DSAs. This value enables HQMUCDSA1 and BOSDSA4 to accept incoming connections over IPv4 and IPv6 protocol.
3. Configure the name service:

   On the machines running HQMUCDSA1 and BOSDSA4, the administrators must configure the name service so that the name resolution process returns all IPv6 and IPv4 addresses. (See your operating system documentation for instructions on how to configure the name service.) The administrators perform the **dirxhostinfo** command to check the name service configuration on both sides. (Both commands should return the IPv4 and IPv6 addresses. The addresses must be the same on both machines unless you have some NAT or proxy relays in between them.)

   The administrators in Munich and Boston perform the commands:

   ```
   dirxhostinfo -I hqmucdsa1
   ```

   The command output is as follows:

   ```
   CanonicalHostname: hqmucdsa1.mycompany.com
   Found HostIP6: 2001:DB8::a0b0:5678
   ```

```
Found HostIP4: 123.45.67.89
```

```
dirxhostinfo -I bosdsa4
```

The command output is as follows:

```
CanonicalHostname: bosdsa4.stu.com
Found HostIP6: 2001:DB8::a012:1234
Found HostIP4: 111.22.33.55
```

The command output shows that the name service is configured correctly because it provides the IPv6 and the IPv4 addresses.

4. The administrators use the following values for the PSAP addresses in the cooperating DSA table (when creating shadowing agreements), the knowledge references, and their own PSAP in **DIRX_OWN_PSAP**:
   - HQMUCDSA1:

     ```
     {TS=DSA1,NA='TCP/IP_IDM!internet=123.45.67.89+port=19999',DNS=HQ
     MUCDSA1+19999:6}
     ```

   - HQMUCDSA2:

     ```
     {TS=DSA2,NA='TCP/IP_IDM!internet=198.76.54.32+port=21111'}
     ```

   - DURDSA3:

     ```
     {TS=DSA3,NA='TCP/IP_IDM!internet=111.22.33.44+port=23333',DNS=DU
     RDSA3+23333}
     ```

   - BOSDSA4

     ```
     {TS=DSA4,NA='TCP/IP_IDM!internet=111.22.33.55+port=24444',DNS=BO
     SDSA4+24444:6}
     ```

The PSAP addresses of the DSAs HQMUCDSA2 and DURDSA3 remain unchanged.These DSAs are not upgraded and run older versions of DirX Directory.The communication from and to these DSAs uses IPv4 protocol.HQMUCDSA2 and DURDSA3 use the IPv4 address set up in the **NA** component of HQMUCDSA1 if their name service does not return a correct

IPv4 address.

The upgraded HQMUCDSA1 and the new BOSDSA4 are able to handle the communication over IPv4 and IPv6 simultaneously.By specifying the IP version **:6** in the PSAP address, the DSA is instructed to use IPv6 communication when establishing an outgoing connection to this partner DSA.Keep in mind that both of these DSAs should set up their correct IPv4 addresses in the **NA** component, because these addresses may be used to communicate over IPv4 protocol.

## 14.6. Encrypting X.500 Protocols

DirX Directory V8.2B and newer supports secure X.500 protocols by using SSL/TLS for the IDM stack.This variant of the communication stack is called IDMS within DirX Directory.All X.500 protocols—DAP, DSP and DISP—can be performed over IDMS between DirX Directory components of V8.2B and newer.

The **DNS** component of a PSAP address controls whether or not SSL/TLS is to be used for a particular X.500 communication.As a result, the internal content of the DNS component has changed in DirX Directory V8.2B (although the old DNS format continues to be supported for compatibility reasons).

To function properly, IDMS needs suitable key material to be installed and configured in the IDMS configuration files.Each entity that uses IDMS has its own configuration file located in the corresponding **conf** folder.For example, for the DSA process (**dirxdsa**), the file is located under *install_path***/server/conf**, for the LDAP server process under *install_path***/ldap/conf** and for **dirxcp** under *install_path***/client/conf**.Note that the DirX Directory installation adds files with suitable key material in *install_path***/conf/IDMPSE.pem** and **testCA.der**.The keys in these files are sufficient to perform IDMS based on the example key material.

The following rules apply for IDMS communication:

- Only the IDM stack provides SSL/TLS functionality.Encrypted X.500 connections are not possible over the OSI stack.
- The **DNS** subcomponent of the “own” address is used to initialize the IDMS security libraries. This address is defined by:
  - A DSA’s **DIRX_OWN_PSAP** environment variable
  - A DUA’s **Self** entry in **dirxldap.cfg** (for the DirX Directory LDAP Server) and **dirxcl.cfg** ( for the **dirxcp** command-line client).

If the **DNS** subcomponent contains an **SSLPORT** subcomponent with a value greater than 0, the IDM stack initializes the openssl libraries (read the key material and so on). In the DSA process, IDMS starts a communication listener on the port specified in **SSLPORT**. The DirX Directory processes can communicate over both variants: plain and encrypted IDM (IDMS). A DSA is can start listeners on two different ports: the port for plain IDM is always activated, the port for encrypted IDMS can optionally be activated.

- To connect to a peer DSA, the initiator of the connection uses the **DNS** subcomponent of the remote DSA’s presentation address and its own initialization state. The remote address is defined by:

    - The PSAP address from a consumer-knowledge attribute for a shadow supplier DSA as an initiator
    - The PSAP address from a supplier-knowledge attribute for a shadow consumer DSA as an initiator
    - The PSAP address from references (**XR**, **SUBR** or **SUPR**) for a DSA in a distributed directory as an initiator
    - The contact DSA entry in **dirxldap.cfg** for the DUA in the LDAP server process
    - The named DSA entries in **dirxcl.cfg** for the DUA in **dirxcp**
    - The **-psap** option in a **dirxcp bind** operation
- Secure X.500 protocols are only supported by DirX Directory components of version 8.2B and newer. However, because the PSAP addresses of all participating DSAs are distributed to all participating DSAs in a shadowing environment, all shadow consumer DSAs in a replication network must support the new **DNS** subcomponent structure introduced for IDMS. Apart from DirX Directory V8.2B and newer, this is the case only for DirX Directory V8.2A with patch level 8.6.351 or newer.

The following example illustrates how to set up and configure a mixed plain IDM and encrypted IDMS communication in an environment with DirX Directory V8.2A (build 8.6.351 or newer) and V8.2B and newer installations.

### 14.6.1. Planning the Encrypted X.500 Configuration

Recall My-Company's network configuration from previous chapters:

- My-Company's headquarters is located in Munich. The site has two DSAs: DSA1's **DNS** component name is **HQMUCDSA1** (My-Company Sales and Development department) and DSA2's **DNS** component name is **HQMUCDSA2**. DSA2 is a shadow consumer of DSA1. (See the chapter "Creating a Shadow DSA" for details.)
- The remote manufacturing plant is located in Durach. The site has one DSA – DSA3 – with the **DNS** component name: **DURDSA3** (My-Company Manufacturing department). (See the chapter "Distributing the DIT across Multiple DSAs" for details.)
- The acquired sales partner, the STU Company in Boston (STU Sales department). This site has one DSA – DSA4 – with the **DNS** component name **BOSDSA4** (See the chapter "Multireplication" for details.)

Headquarters in Munich wants to use the encrypted variant of the X.500 DISP replication protocol (IDMS) to BOSDSA4. Updates chained from the consumer BOSDSA4 to the supplier DSA HQMUCDSA1 via X.500 DSP are also required to be encrypted. Recall from the chapter "Multireplication" that the My-Company and STU companies decide to replicate My-Company's Sales and Development employee directory on the DSA in Boston, and to replicate the STU Company's employee directory on the DSA in Munich. While the connections between DSA1 (HQMUCDSA1) in Munich and DSA4 (BOSDSA4) in Boston will use IDMS, the connections between DSA1 and DSA2 in Munich and DSA3 in Durach will run over plain IDM. The following figure illustrates this configuration:

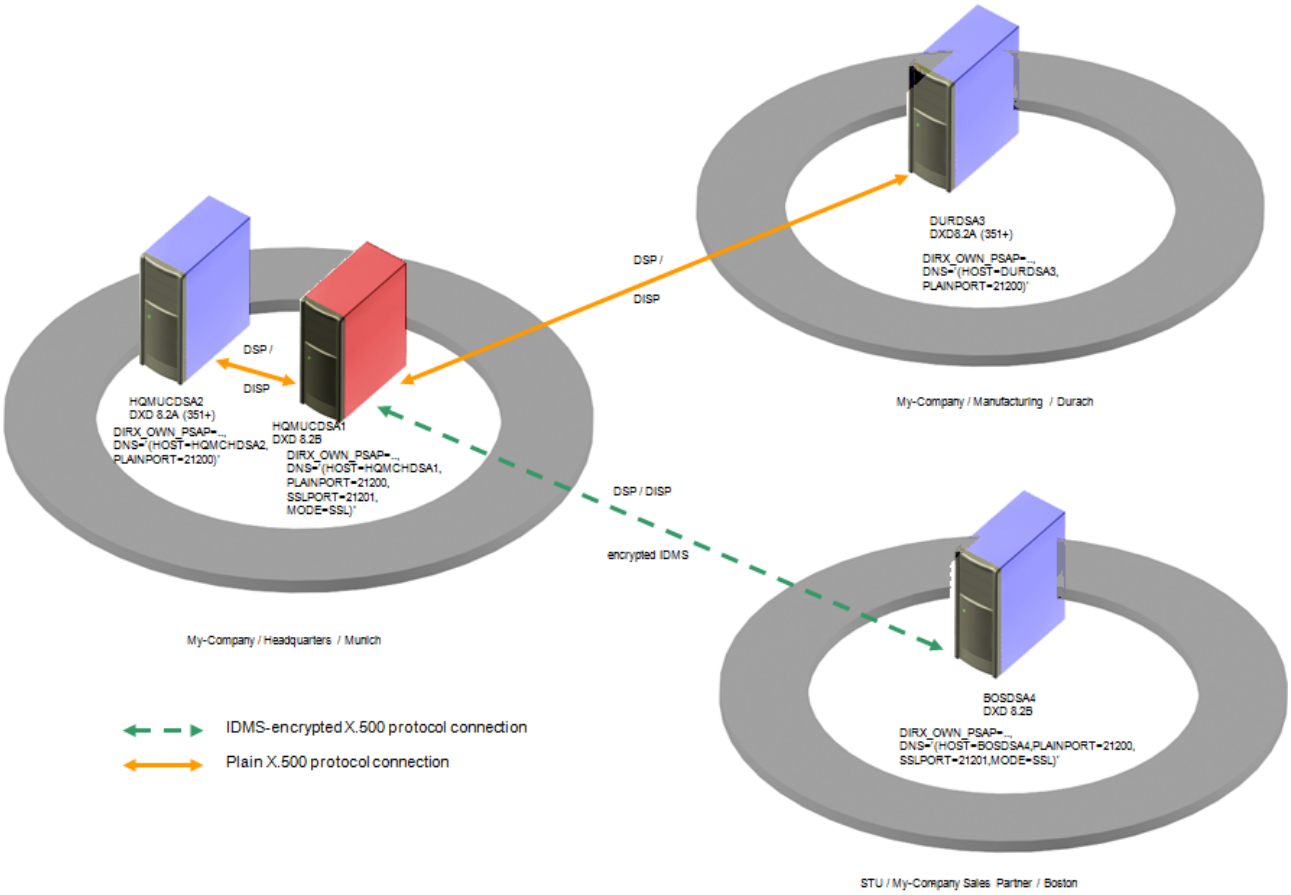



*Figure 41. Network with Mixed IDM and IDMS Communication*

In the figure above, all DSAs contain the following values in their cooperating DSA attribute:

```
SUK: PSAP= ...,
DNS='(HOST=HQMCHDSA1,PLAINPORT=21200,SSLPORT=21201,MODE=SSL)'
CK1: PSAP=...,
DNS='(HOST=BOSDSA4,PLAINPORT=21200,SSLPORT=21201,MODE=SSL)'
CK2: PSAP=... ,DNS='(HOST=MUCDSA2,PLAINPORT=21200)'
CK3: PSAP=... ,DNS='(HOST=DURDSA3,PLAINPORT=21200)'
```

## 14.6.2. Setting up the Encrypted X.500 Configuration

To set up this configuration, the administrators must:

- Upgrade the DirX Directory software on the DSAs to support IDMS
- Set up the DSA presentation addresses
- Create the shadowing agreements

The next sections describe these steps.

#### 14.6.2.1. Upgrading the DirX Directory Software on the DSAs

The first step is for the administrators of the DSAs in Munich, Boston and Durach to upgrade their DSAs to DirX Directory software that supports IDMS:

- The administrators of HQMUCDSA1 in Munich and BOSDSA4 in Boston upgrade their systems to DirX Directory V8.2B or newer.
- The administrators of HQMUCDSA2 in Munich and DURDSA3 in Durach ensure that these DirX Directory V8.2A installations are upgraded to patch level 8.6.351 or newer to enable the DSAs on these hosts to be able to parse the **DNS** component syntax for IDMS communication contained in the PSAP addresses (keywords like **HOST**, **SSLPORT**, **PLAINPORT**, **MODE** and so on). The shadow supplier DSA distributes these PSAP addresses to all shadow consumer DSAs as part of the supplier and consumer knowledge values contained in the cooperating DSAs subentry. See the section **Presentation-Address** in the *DirX Directory Syntaxes and Attributes* for a complete description of **DNS** component syntax for IDMS.

#### 14.6.2.2. Setting up the DSA Presentation Addresses

Next, the administrators set up each DSA's **DIRX_OWN_PSAP** environment variable. DSA1 in Munich and DSA4 in Boston are to perform encrypted shadowing protocol.

On HQMUCDSA1 in Munich, the administrator sets the **DIRX_OWN_PSAP** environment variable to:

```
TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=1234',
DNS='(HOST=HQMUCDSA1,PLAINPORT=21200,SSLPORT=21201,MODE=SSL)'
```

On BOSDSA4 in Boston, the administrator sets the **DIRX_OWN_PSAP** environment variable to:

```
TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=1234',
DNS='(HOST=BOSDSA4,PLAINPORT=21200,SSLPORT=21201,MODE=SSL)'
```

These two values activate IDMS in the DSAs on HQMUCDSA1 and BOSDSA4 after the DSA processes are restarted. The DSAs on these hosts will start a listener for plain X.500 protocols over IDM on port 21200 and a listener for encrypted X.500 protocols over IDMS on port 21201 so that they are prepared to accept incoming secure X.500 protocols. Note that in DirX Directory, only the **DNS** component of the PSAP is used for addressing if it is contained in the PSAP address. The **TS** and **NA** components are ignored, but must be present due the X.500 attribute specification of the PSAP address. This is the reason that these examples have the **NA** value set to **1.2.3.4** with port **1234**.

On HQMUCDSA2 in Munich and DURDSA3 in Durach, the **DIRX_OWN_PSAP** environment variable must not contain an **SSLPORT** subcomponent.

#### 14.6.2.3. Creating the Shadowing Agreements

Now the administrator in Munich sets up the shadowing agreements on the shadow supplier DSA HQMUCDSA1.

First, he creates the encrypted shadow agreement to the consumer DSA in Boston with the **dirxadm sob create** command. In the **supplierpsap** option, he supplies the following value:

```
TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=1234',
DNS='(HOST=HQMUCDSA1,PLAINPORT=21200,SSLPORT=21201,MODE=SSL)'
```

In the **consumerpsap** option, he supplies the following value:

```
TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=1234',
DNS='(HOST=BOSDSA4,PLAINPORT=21200,SSLPORT=21201,MODE=SSL)'
```

These values result in encrypted X.500 DISP and DSP exchanges between the supplier and the consumer DSA as follows:

- By specifying **SSLPORT** in the supplier PSAP address, all consumer DSAs receive the knowledge that the supplier DSA is listening on both IDM and IDMS ports.
- By specifying **MODE=SSL** in the supplier PSAP address, all consumer DSAs receive the knowledge that the supplier DSA prefers communicating over IDMS. Therefore, a consumer DSA that supports IDMS (BOSDAS4 in the example) will connect to the IDMS port of the supplier DSA when initiating X.500 DSP connections for chaining.
- By specifying **MODE=SSL** in the consumer PSAP address, the supplier DSA will connect to the IDMS port when initiating X.500 DISP connections.

The creation of this shadowing agreement - and any new shadowing agreements subsequently created on the supplier DSA - automatically distributes the updated cooperating DSAs subentry containing the DNs and the PSAPs of all DSAs that form the replication network to all DSAs listed as consumer DSAs in the subentry.

Next, the Munich DSA1 administrator creates the plain shadow agreements to the consumer DSAs in Munich headquarters (DSA2) and in Durach (DSA3).

In the **supplierpsap** option, he supplies the following value:

```
TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=1234',
DNS='(HOST=BOSDSA4,PLAINPORT=21200,SSLPORT=21201,MODE=SSL)'
```

The **supplierpsap** option can be omitted from the **sob create** command because the PSAP has already been associated with the supplier DSA's distinguished name in the first shadowing agreement)

In the **consumerpsap** option, he supplies the following value:

```
TS=DSA1,NA='TCP/IP_IDM!internet=1.2.3.4+port=1234',
DNS='(HOST=MUCDSA2,PLAINPORT=21200)'
```

These values have the following effects:

- The supplier DSA will connect to the IDM port when initiating X.500 DISP connections.
- All consumer DSAs that do not support IDMS will ignore the specification of **SSLPORT** and **SSL** for **MODE** in the supplier's PSAP address. These consumer DSAs (HQMUCDSA2 and DURDSA3 in the example) will connect to the IDM port of the supplier DSA when initiating X.500 DSP connections.

All consumer DSAs in the replication network, regardless of whether they can support IDMS, must be able to parse the **DNS** component syntax for IDMS communication specified in the PSAP addresses/knowledge references given in the replicated cooperating DSA subentry. This is the reason why all DirX Directory installations that form a replication network must be at patch level 8.6.351 or newer.

### 14.6.3. Switching Masters in the Encrypted X.500 Configuration

The DSA on host HQMUCDSA2 in the My-Company scenario can act as a standby master DSA: it has a full copy of the shadow supplier's directory data and can be made the shadow supplier with the **dirxadm sob switch** operation. See the section "Switching Masters in a Running System" in the chapter "Creating a Shadow DSA" for details.

Such a switch operation results in performing the replication to BOSDSA4 as plain X.500 DISP. If this is unacceptable for My-Company, the installation of HQMUCDSA2 must be upgraded to DirX Directory 8.2B or newer.

### 14.6.4. Creating Customized Key Material

The DirX Directory installation creates IDMS configuration files and key material files that are sufficient to perform IDMS.However, if My-Company decides to use key material issued by its own Certification Authority to secure the DISP and DSP exchanges between HQMUCDSA1 and BOSDSA4, the following steps are required:

- My-Company's CA issues a PEM-formatted file with key material for each of the DSAs.These files contain the respective DSA's private key, the DSA's public key certificate and optionally the root CA certificate (see the installed key material file in *install_path***/server/conf/dirxIDM.pem** as a template).These files are password-protected.
- Copy the key material file to the machine and then configure the pathname in the **idmssl.cfg** file located in *install_path***/server/conf/idmssl.cfg**.Enter the pathname of the key material file after the keyword **idm_ssl_own_pse_file**.

- Create a new file, write the protection password into it and then configure the password file in the **idmssl.cfg** file located in *install_path***/server/conf/idmssl.cfg**.Enter the pathname of the password file after the keyword **idm_ssl_pwd_file**.The password will be encrypted by the DSA after first use.
- Copy the PEM-formatted root CA certificate of all CAs that issued key material for all communication peers to a trusted CA file on the machine.In this example, My-Company uses the same CA as the issuer for the keys for both peers; therefore, the trusted CA files on both machines must contain only this one Root CA certificate.Configure the Root CA Certificate pathname in the **idmssl.cfg** file located in *install_path***/server/conf/idmssl.cfg**.Enter the pathname of the Root CA file after the keyword **idm_ssl_trusted_ca_cert_file**.

# 14.7. Changing Host IP Addresses of DirX Directory Servers

Changes to DirX Directory server host IP addresses – such as using a different DNS host name or using a different IP address or port number for a given host – that result from moving DirX Directory databases to new hardware or restructuring the network require administrative action to get the DirX Directory service up and running again in the new IP configuration.

The following elements of a DirX Directory installation contain DNS hostnames, IP addresses, and port numbers as PSAP addresses.These elements must be updated with new DNS names or target IP addresses should a change to a DirX Directory server host IP address become necessary:

- The **DIRX_OWN_PSAP** environment variable
- The LDAP server configuration file **$DIRX_INST_PATH/ldap/conf/dirxldap.cfg**
- The **dirxcp**/**dirxadm** configuration file **$DIRX_INST_PATH/client/conf/dirxcl.cfg**
- The My-Access-Point (MAC) attribute of the root DSE (**/**) (database content)
- Supplier and consumer PSAP addresses in shadowing agreements (database content)
- Reference entries (**SUBR**, **SUPR**, ...) in a distributed DIT configuration (database content)

The *DirX Directory Administration Reference* and the *DirX Directory Syntaxes and Attributes* documents provide detailed information about these elements.

The next sections describe procedures for updating these elements in different DirX Directory service deployment scenarios. Before performing any of these procedures, a database backup should be created for all affected DSAs and the generated backup files should be *verified* and *error free*. See the description of the **dirxbackup** command in the *DirX Directory Administration Reference* for details on how to perform database backups.

Use DNS format as much as possible in PSAP addresses to make IP address changes easier in the future. When DNS format is used, the DirX Directory service calls the DNS resolver instead of using a hard-coded IP address.

The PSAP addresses of any reference entries (**SUBR**, **SUPR**, ...) that are present in the

database should also be updated with the new IP address information. This task can be performed any time since these entries are not affected by the procedures described below.

### 14.7.1. Changing IP Addresses in a Standalone DirX Directory Installation

To update a standalone DirX Directory installation with a changed IP address:

- Stop the DirX Directory service. (See **dirxadm sys stop** in the *DirX Directory Administration Reference* for details.)
- Start the **dirxdsa** process only, in administrative mode: call **dirxdsa -a** from a command shell.
- Run the **dirxadm remove_knowledges** command (this command deletes all LOB and SOB agreements). This is a special command and only used in this case.
- Stop the **dirxdsa** process: type <Ctrl>+C in its command shell.
- Modify the **DIRX_OWN_PSAP** environment variable to add the new PSAP address with the new IP address.
- Modify the LDAP server and client configuration files to add the new PSAP address with the new IP address.
- Update the network configuration in the operating system to change to the required new host IP address.
- Start the DirX Directory service as usual. (See **dirxadm sys start** in the *DirX Directory Administration Reference* for details.)
- Run the **dirxadm update_own_PSAP** command to update the My-Access-Point (MAC) attribute of the root DSE (**/**). (This attribute is not automatically updated when the **DIRX_OWN_PSAP** environment variable is changed.) The **dirxadm update_own_PSAP** is a special command and only used in this case.
- Check the My-Access-Point (MAC) attribute of the root DSE (**/**) by running the **dirxadm show / -attr MAC** command. The output should show a PSAP address that is identical to the **DIRX_OWN_PSAP** environment variable.

### 14.7.2. Changing Supplier IP Addresses in a Shadowing Scenario

To update a shadowing scenario when only the *supplier* IP address is changed:

- Save all shadowing agreement data on the *supplier*.
- Terminate and delete *all* shadowing agreements to *all* consumers on the *supplier*. Consumer DSAs must be up and running during these operations.
- Perform all the steps described in the section "Changing IP Addresses in a Standalone DirX Directory Installation" on the *supplier* DSA. The supplier now has the new host IP configuration.
- Recreate *all* necessary shadowing agreements with the new supplier PSAP address.

### 14.7.3. Changing Consumer IP Addresses in a Shadowing Scenario

To update a shadowing scenario when only a *consumer* IP address is changed:

- Save the affected shadowing agreement data on the *supplier*.
- Terminate and delete the shadowing agreements to this consumer on the *supplier*. The selected consumer must be up and running during these operations.
- Stop the DirX Directory service on the selected consumer.
- Modify the **DIRX_OWN_PSAP** environment variable on this consumer to add the new PSAP address with the new IP address.
- Modify the LDAP server and client configuration files to add the new PSAP address with the new IP address.
- Run **dbamboot** to reset the database on this consumer.
- Update the network configuration in the operating system to change to the required new host IP address. The consumer now has the new host IP configuration and an empty database.
- Start the DirX Directory service on the consumer.
- Recreate *all* necessary shadowing agreements with the new consumer PSAP address.

### 14.7.4. Changing Supplier and Consumer IP Addresses in a Shadowing Scenario

To update a shadowing scenario when *both* supplier *and* consumer IP addresses are changed:

- Save all shadowing agreement data on the *supplier*.
- Terminate and delete *all* agreements to *all* consumers on the *supplier*.All consumers must be up and running during these operations.
- Stop the DirX Directory service on the consumer(s).
- Modify the **DIRX_OWN_PSAP** environment variable on the consumer(s) to add the new PSAP address with the new IP address.
- Modify the LDAP server and client configuration files on the consumer(s) to add the new PSAP address with the new IP address.
- Run **dbamboot** to reset the database on the consumer(s).
- Update the network configuration in the operating system to change to the required new host IP address.The consumer(s) now has the new host IP configuration and an empty database.
- Start the DirX Directory service on the consumer(s).
- Perform all the steps described in the section "Changing IP Addresses in a Standalone DirX Directory Installation" on the *supplier*.The supplier now has the new host IP configuration.
- Recreate *all* the necessary shadowing agreements with the new PSAP address.

# 14.8. Using Two-Factor Authentication (2FA)

Two-factor authentication (2FA) enhances service security by requiring two forms of identification for user access to a service.DirX Directory implements the time-based one-time password (TOTP) type of two-factor authentication for DirX Directory users.

In a TOTP-based 2FA DirX Directory configuration, TOTP 2FA-enabled DirX Directory users are expected to supply their passwords plus a 6-digit TOTP issued by a TOTP authenticator app on the user side when binding to a TOTP 2FA-configured DirX DSA.The user TOTP is based on the current time and a secret key ("TOTP secret") shared between the DSA and the user. This TOTP secret is created by the DSA when a user is enabled for TOTP 2FA and is given to the user by the DirX Directory administrator because the user is not allowed to read/view the dirxTOTPSecret attribute from his own entry. The user in turn loads the received TOTP secret into the TOTP authenticator app, allowing the app to generate TOTPs for this user for TOTP 2FA authentication to the DirX Directory service.The authenticator app periodically updates the TOTP to be used for authentication; usually, every 30 seconds.

The next sections describe how to set up and manage the DirX Directory TOTP 2FA environment.For general information about TOTP 2FA, see RFC 6238: TOTP: Time-Based One-Time Password Algorithm.

## 14.8.1. Configuring the DirX DSA for TOTP 2FA

The steps to set up the DSA for TOTP 2FA are:

- Configure the access control items (ACIs) for TOTP 2FA administrators to allow read/write access to TOTP 2FA-related object classes and attributes.
- Ensure that the DSA's TOTP validity period is the same as the TOTP validity period used by the authenticator app on the user side and adjust it on the DSA with the **DIRX_DSA_TOTP_PERIOD** environment variable as necessary.
- Ensure that the clocks on the DirX DSA machine and TOTP 2FA user devices are closely synchronized (recommendation: to within 1-2 seconds difference), preferably via NTP server. For a 30-second TOTP validity period, even a 15-second difference between clocks can make TOTP 2FA binds impossible.
- Use the environment variable **DIRX_DSA_TOTP_ISSUER** to specify the name of the provider associated with DSA TOTP secret generation if it is different from the default string **DirX**.

The next sections provide more detail.

#### 14.8.1.1. Configuring TOTP 2FA Administrator ACIs

DirX Directory administrator(s) (DAP and LDAP) are responsible for enabling and disabling TOTP 2FA for DirX Directory users. To carry out their TOTP 2FA administration tasks, these administrators need read/write access to two TOTP 2FA-specific items in a user's entry:

- The **dirxTOPUser** object class, which the administrators add/remove to a user entry to enable/disable TOTP 2FA for the user

- The **dirxTOPSecret** attribute, which the DSA creates and populates with a TOTP secret when a user is 2FA-enabled and which the administrator must read to get the TOTP secret and then send to the user

As a result, the ACI settings for DirX Directory administrators involved in TOTP 2FA management must be configured to allow read/write access to these items for all user entries. These administrators should be the only entities permitted this kind of access to these items.

The My-Company example database already contains the necessary access rights for the **cn=admin,o=my-company** user to configure TOTP 2FA for other users. Suppose this administrator wants to create a separate TOTP 2FA administrator **cn=admin2,o=my-company**) and grant this administrator the necessary access rights for TOTP 2FA administration. To do this, he performs the following steps over DAP protocol:

- Creates the user /o=my-company/cn=admin2 with dirxcp:

```
dirxcp> create /o=My-Company/cn=admin2 -attr OCL=ORP
SN=admin2 UP=dirx
```

- Adds a new ACI item to the Prescriptive ACI attribute of the relevant Access Control Subentry **(/o=My-Company/cn=AccessControl-Subentry**) for the previously created **admin2** entry:

```
dirxcp> modify /o=My-Company/cn=AccessControl-Subentry
-add {PACI={ID=Admin2: TOTP 2FA administrator rights,
       PR=254,
       AL={BL={L=SIMPLE}},
       UF={UC={N={DN={/O=My-Company/CN=admin2}}},
       UP={PI={E=TRUE}, GAD=grantModify};
      {PI={AT=OCL;TOTPS,AAV=OCL;TOTPS},
           GAD=grantRead+grantAdd+grantRemove}}}}
```

#### 14.8.1.2. Synchronizing the TOTP Period with the Authenticator App

Most TOTP authenticator apps change a user's TOTP every 30 seconds. As a result, DirX Directory uses a default value of 30 seconds and specifies this default in the DirX environment variable **DIRX_DSA_TOTP_PERIOD**. If a TOTP authenticator in a DirX Directory TOTP 2FA scenario uses a different refresh period, you must modify the **DIRX_DSA_TOTP_PERIOD** environment variable to match the period used by the authenticator app.

#### 14.8.1.3. Changing the TOTP Secret Issuer Name

The format for the TOTP secret used by the DirX DSA and recognized by most TOTP authenticator apps includes a parameter for specifying the provider associated with the

TOTP secret. TOTP authenticator apps typically display this parameter along with the TOTP displayed for the user. By default, the DSA supplies the value **DirX** in this parameter when it creates the secret. To have the DSA supply a different value for subsequent display by an authenticator app, specify it in the environment variable **DIRX_DSA_TOTP_ISSUER**. For example, to have a TOTP authenticator app display **My-Company** along with user TOTPs for binding to a DSA, set the DSA's **DIRX_DSA_TOTP_ISSUER** environment variable as follows:

```
DIRX_DSA_TOTP_ISSUER=My-Company
```

### 14.8.2. Setting up a QR Code Generator (Optional)

Administrators need to send the TOTP secret generated by the DSA to the 2FA-enabled user for loading into the user's authenticator app. The app needs to have the user's TOTP secret to be able to generate TOTPs for the user for binding to the DirX Directory service.

Most authenticator apps recognize the TOTP secret format used by the DSA and users can copy and paste this format into these apps without problems. However, we recommend setting up a QR code generator for converting DSA-generated TOTP secrets into QR codes for delivery to TOTP 2FA users. Although DirX Directory does not include a built-in QR code generator, there are many third-party options for generating QR codes available for use, and many of these options are free of charge.

### 14.8.3. Enabling TOTP 2FA for a User

To enable TOTP 2FA for a user, use the **dirxcp** [**obj**] **modify** command with the **-add** option to add the object class **dirxTOTPUser** to the user entry over DAP or LDAP. For example, the following command adds the object class **dirxTOTPUser** to the user entry **cn=Tinker,ou=Sales,o=My-Company** over LDAP:

```
dirxcp> modify cn=Tinker,ou=Sales,o=My-Company
-add objectClass=dirxTOTPUser
```

See the description of the **dirxcp** command in the *DirX Directory Administration Reference* for details on **dirxcp** command syntax and options.

When the DSA receives the modify request, it creates the attribute **dirxTOTPSecret** with a 32 byte, cryptographically-secure random secret value for the specified user. This value is encrypted in the database in AES-256 format.

### 14.8.4. Distributing the TOTP Secret to the User

Once you have enabled a user for TOTP 2FA, this user can no longer perform a simple bind with only his user password. Hence it is vital to share a user's TOTP secret with them as soon as possible.

First, use the **dirxcp** [**obj**] **search** command to read the user's **dirxTOTPSecret** attribute from the directory over DAP/LDAP. For example, over LDAP:

```
dirxcp> search cn=Tinker,ou=Sales,o=My-Company -base
-attr dirxTOTPSecret
```

This action returns the user's TOTP secret decrypted into the format recognized by most authenticator apps (see the description of the DirX TOTPSecret attribute in the section "X.500 User Application Attributes" in the chapter "DirX Directory Attributes" in *DirX Directory Syntaxes and Attributes* for details on this format). For example, the result of the LDAP request above is:

```
{cn=Tinker,ou=Sales,o=My-Company
dirxTOTPSecret=otpauth://totp/cn=Tinker,ou=Sales,o=My-
Company?issuer=DirX&secret=MGHBFTTMOWTS6FEKJ5PDXRXMAMRL2WJXA3SB2CJOPG
QASD6GO4RQ====}
```

Now you can either send the secret to the user (for example, by email) in the format returned by the DSA, or (recommended) use the QR code generator you previously set up to convert the secret into a QR code and then send the QR code to the user.

The user then pastes or scans the received secret into the authenticator app, enabling it to generate TOTPs for the user and thus enabling the user to bind to the DirX Directory service with TOTP 2FA.

### 14.8.5. Binding with TOTP 2FA

A user can perform a TOTP 2FA bind with DAP or with LDAP. The password option format for a TOTP 2FA bind is:

**-password** *password*,*totp*

where *password* is the user's password for simple binds and *totp* is the current TOTP that a TOTP authenticator app has generated for the user. For example, suppose the user **cn=Tinker,ou=Sales,o=My-Company** has the password **dirx** and the current TOTP issued by the authenticator app is **123456**. The bind command over LDAP is:

```
dirxcp> bind -u cn=Tinker,ou=Sales,o=My-Company -proto ldapv3
-auth simple -password dirx,123456
```

The bind command over DAP is:

```
dirxcp> bind -user /O=My-Company/OU=Sales/CN=Tinker
-auth simple -password dirx,123456
```

See the description of the **dirxcp** command in the *DirX Directory Administration Reference*

for details on command syntax and options.

Reusing a TOTP for a bind operation is not allowed; once a successful TOTP 2FA bind is made, the user must wait until the authenticator app displays a new TOTP.

Once a user makes a successful TOTP 2FA bind, they are authenticated at the strong ACI level.

### 14.8.6. Disabling TOTP 2FA for a User

To disable TOTP 2FA for a user, use the **dirxcp** [**obj**] **modify** command with the **-rem** option to remove the object class **dirxTOTPUser** from the user entry over DAP or LDAP. For example, the following command removes the object class **dirxTOTPUser** from the user entry **cn=Tinker,ou=Sales,o=My-Company** over LDAP:

```
dirxcp> modify cn=Tinker,ou=Sales,o=My-Company
-rem objectClass=dirxTOTPUser
```

The DSA automatically removes the **dirxTOTPSecret** attribute.

See the description of the **dirxcp** command in the *DirX Directory Administration Reference* for details on command syntax and options.

### 14.8.7. Generating a New TOTP Secret for a User

To generate a new secret for a user, remove the object class **dirxTOTPUser** and then add it again, as described earlier in this section. Be sure to send the new secret to the user after the DSA generates it.

### 14.8.8. Troubleshooting TOTP 2FA Bind Failures

Here are some common issues with TOTP 2FA bind failures and some possible solutions:

- User TOTP 2FA configuration problems: Check the configured user password and TOTP secret for the user.
- Clock synchronization problems: Make sure that the time difference between the user's device and the server machine is very small. Recommendation is 1-2 seconds.
- Problems with a TOTP secret: Try generating a new secret for the user by removing the object class **dirxTOTPUser** and then adding it again (see the sections above for details). Do not forget to distribute this new secret to the user.
- User error: Check to make sure the user tried to authenticate using a TOTP. The user may have forgotten that they are a TOTP 2FA user and tried to authenticate without a TOTP.
- TOTP period mismatch: Make sure the TOTP validity period in the user's TOTP authenticator app is the same as the period specified in the DSA. Note that most TOTP authenticator apps only support a 30-second TOTP period.

- Password option syntax error: Make sure the syntax used to supply the TOTP 2FA password option to **dirxcp** is correct (see the section on binding with TOTP 2FA for an example). See also the **dirxcp** [**obj**] **bind** operation description in the *DirX Directory Administration Reference* for details on command syntax and examples.
- User TOTP 2FA configuration problems: Make sure that the user has both the object class **dirxTOTPuser** and the attribute **dirxTOTPSecret**.

# DirX Product Suite

The DirX product suite provides the basis for fully integrated identity and access management; it includes the following products, which can be ordered separately.

## DirX Identity

DirX Identity provides a comprehensive, process-driven, customizable, cloud-enabled, scalable, and highly available identity management solution for businesses and organizations. It provides overarching, risk-based identity and access governance functionality seamlessly integrated with automated provisioning. Functionality includes lifecycle management for users and roles, cross-platform and rule-based real-time provisioning, web-based self-service functions for users, delegated administration, request workflows, access certification, password management, metadirectory as well as auditing and reporting functionality.

## DirX Directory

DirX Directory provides a standards-compliant, high-performance, highly available, highly reliable, highly scalable, and secure LDAP and X.500 Directory Server and LDAP Proxy with very high linear scalability. DirX Directory can serve as an identity store for employees, customers, partners, subscribers, and other IoT entities. It can also serve as a provisioning, access management and metadirectory repository, to provide a single point of access to the information within disparate and heterogeneous directories available in an enterprise network or cloud environment for user management and provisioning.

## DirX Access

DirX Access is a comprehensive, cloud-ready, scalable, and highly available access management solution providing policy- and risk-based authentication, authorization based on XACML and federation for Web applications and services. DirX Access delivers single sign-on, versatile authentication including FIDO, identity federation based on SAML, OAuth and OpenID Connect, just-in-time provisioning, entitlement management and policy enforcement for applications and services in the cloud or on-premises.

## DirX Audit

DirX Audit provides auditors, security compliance officers and audit administrators with analytical insight and transparency for identity and access. Based on historical identity data and recorded events from the identity and access management processes, DirX Audit allows answering the “what, when, where, who and why” questions of user access and entitlements. DirX Audit features historical views and reports on identity data, a graphical dashboard with drill-down into individual events, an analysis view for filtering, evaluating, correlating, and reviewing of identity-related events and job management for report generation.

For more information: support.dirx.solutions/about

Legal remarks

On the account of certain regional limitations of sales rights and service availability, we cannot guarantee that all products included in this document are available through the Eviden sales organization worldwide. Availability and packaging may vary by country and is subject to change without prior notice. Some/All of the features and products described herein may not be available locally. The information in this document contains general technical descriptions of specifications and options as well as standard and optional features which do not always have to be present in individual cases. Eviden reserves the right to modify the design, packaging, specifications and options described herein without prior notice. Please contact your local Eviden sales representative for the most current information. Note: Any technical data contained in this document may vary within defined tolerances. Original images always lose a certain amount of detail when reproduced.