Tivoli. software

IBM

Performance Tuning for IBM Tivoli Directory Server

Performance tuning for Tivoli Directory Server for very large user environments

Complete coverage from operating system to database tuning

Extensive scripts and checklists

Axel Buecker
Robert Hodges
Richard Macbeth
John McGarvey
Casey Peel
Jukka Rantanen

Redpaper

ibm.com/redbooks

International Technical Support Organization

Performance Tuning for IBM Tivoli Directory Server

March 2007

Note: Before using this information and the product it supports, read the information in “Notices” on page ix.

First Edition (March 2007)

This edition applies to different versions of IBM Tivoli Directory Server and its underlying support software.

© Copyright International Business Machines Corporation 2007. All rights reserved.
Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

© Copyright IBM Corp. 2007. All rights reserved.

Notices

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

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

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

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

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

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

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

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

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

COPYRIGHT LICENSE:

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

Trademarks

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

AIX 5L™	HACMP™	Redbooks™
AIX®	IBM®	Tivoli®
DB2 Universal Database™	RDN™	WebSphere®
DB2®	Redbooks (logo) ™	

The following terms are trademarks of other companies:

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

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

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

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

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

Preface

In today's highly connected world, directory servers are the IT cornerstone of many businesses. These components of the corporate infrastructure are the foundation of authentication systems for internal, and more commonly, external user populations. Managing a directory server with several hundred internal users is not all that difficult. However, when managing a directory server with several million external users in all 24 time zones throughout the world is a much more daunting task.

IBM® Tivoli® Directory Server is capable of handling millions of entries given the right architecture, configuration, and performance tuning—tunings that can differ greatly from that of a smaller server with only a few hundred thousand entries. Managing and tuning a directory server of this size requires a change in mindset: No longer can tuning be done after the fact. Tuning and performance must be a focus before the hardware is even ordered. A proactive role must be taken after installation as well, including pre-tuning steps to better interface with other products to make installations and migrations more successful, and regular maintenance to keep the directory well-tuned and running smoothly.

This IBM Redpaper is the cumulation of lessons learned in many different real-world environments, including a 24-server fault tolerant configuration with over 300 million entries. The authors have pooled their collective knowledge and resources to provide the most comprehensive performance view possible, from hardware to software, sort heaps to buffer pools, and table cardinalities to explain plans.

In large directory server deployments, use this document as an outline on how to get the right fit for your environment.

The team that wrote this IBM Redpaper

This IBM Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Austin Center.

Axel Buecker is a Certified Consulting Software IT Specialist at the ITSO, Austin Center. He writes extensively and teaches IBM classes worldwide on areas of Software Security Architecture and Network Computing Technologies. He holds a degree in computer science from the University of Bremen, Germany. He has 20 years of experience in a variety of areas related to Workstation and Systems Management, Network Computing, and e-business Solutions. Before joining the ITSO in March 2000, Axel worked for IBM in Germany as a Senior IT Specialist in Software Security Architecture.

Robert Hodges is a Systems Management Architect working for the Tivoli Services organization. He has 27 years of experience in systems management and IT architecture with the last 16 of those focused on the Tivoli product set and its use. Bob has multiple patents and has written or coauthored numerous papers and best-practices guides on systems management disciplines.

Richard Macbeth is an IBM Directory Services Architect for IBM Software Services Tivoli, Services Delivery, Americas Security Practice. He has been with IBM for 27 years in the computer/IT field with 15 years of experience in the Lightweight Directory Access Protocol (LDAP) directory field. He has current certifications with Novell as a Certified Directory Engineer, Certified Novell Instructor, Certified Novell Engineer, and Sun™ One Directory 5 Engineer. He has worked on SecureWay/IBM Directory Server on most platforms for seven years and also has seven years of experience with Tivoli Access Manager for e-business. He also held a CCNP Certification with Cisco and had over 10 years of experience as a Senior Network IT Specialist. He has coauthored two IBM Redbooks™, the last one: *Understanding LDAP - Design and Implementation*, SG24-4986, in June 2004. He holds an AS degree in Psychology and a Computer Technology Certification.

John McGarvey is a senior technical staff member for Tivoli at Research Triangle Park in North Carolina. For the past five years he has been the product architect for Tivoli Directory Server and has contributed to the design of numerous scalability and performance enhancements for this product. He also has expertise in TCP/IP internals, IBM DB2® tuning, and security management products.

Casey Peel is an Advisory Software Engineer for Tivoli specializing in performance tuning for IBM Tivoli security products. He joined IBM Austin upon graduation from Texas A&M University with a BS in Computer Science in 2000. Casey works with IBM Tivoli Identity Manager and its middleware: IBM Tivoli Directory Server, IBM WebSphere® Application Server, and IBM DB2 Universal Database™. His work on Identity Manager includes maintaining the IBM Tivoli Identity Manager tuning guides, educating internal and external audiences about performance issues, and supporting customers around the world.

Jukka Rantanen is a Senior IT Specialist in Global Technology Services in Helsinki, IBM Finland. Jukka has worked with DB2 since 1988 and joined IBM in 2000. He has graduated in the Department of Computer Science at the Helsinki University. Jukka is a Database Team leader and he supports large Finnish customer DB2 deployments focusing on performance tuning, high available solutions, and so on.

We like to thank the following people who contributed to making this document.

A special thanks to Anne for her help and expertise with DB2 and for thinking out of the box for more places to look to tweak DB2.

Anne Lesell, IBM Finland IT Specialist with DB2 Information Technology Services, IT Services.

A special thanks to Karen for her assistance with information for using DB2 on SAN arrays. This information was taken from her document called: *Best Practices for EMC CLARiiON CX600 with IBM DB2 Universal Database*, published May 24, 2004 IBM.

Karen Duxbury, IBM Canada Ltd., IBM Toronto Lab.

A special thanks to Michael for rewriting some of the scripts that we have included in this document and to Jeff for helping to convert some of them to be used on a Windows® base platform.

Michael Seedorff, IBM USA IT Specialist IBM software Group ISST, Americas Security Practice.

Jeff DeMent, IBM USA Tivoli Directory/Security Architect IBM software Group ISST, Americas Security Practice.

A special thanks for the following people who put together the online backup section for IBM Tivoli Directory Server.

Cindy Corn, USA Software Engineer, LDAP Development, IBM Software Group, Tivoli.

Shevaun M. Fontenot, USA Software Engineer, LDAP Development, IBM Software Group, Tivoli.

John D. Sullivan, USA Software Engineer, Security Architect, IBM Software Group, Tivoli.

Mark McConaughy, USA Software Engineer, LDAP Development, IBM Software Group, Tivoli.

Shirley Chrisco, USA Software Engineer, Information Development, IBM Software Group, Tivoli.

A special thanks to Chunhui for her inputs on the checklist.

Chunhui Yang, USA Certified Sir ITS, IBM Software Group ISST, Americas Security Practice.

A special thanks to Ram for his assistance with IBM AIX® tuning and reviewing this document.

Ram Sreerangam, USA Security Consultant, IBM Software Group ISST, Americas Security Practice.

And a very special thanks to our families who have given up a lot, for us to do our jobs; we would not have been able to do this without their help and support.

Become a published author

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

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

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

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

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

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbooks@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HYTD Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400

1

Service level objectives and agreements

Every component within an IT environment should have an overall performance objective, an agreement, or both, such as a *service level objective* (SLO) or a *service level agreement* (SLA). These objectives and agreements should be used in both the initial design (hardware, network, redundancy, placement, monitoring, management, and so on) and the ongoing maintenance and measurement of the service to determine if it is still meeting its objectives and agreements.

With IBM Tivoli Directory Server, like many applications that are transaction based, it can be difficult to predict its operation and performance on paper. Until it is configured, the data is loaded and the actual client transactions applied, only rules of thumb can be used. The size of the directory objects (fat versus thin), transaction mix (number of clients, adds, modifies, searches, and so on) and the types and quantities of results that must be returned to the clients can vary significantly throughout its service life.

In addition, because the Directory Server is normally a data store/back-end for user-facing applications, setting baselines (service level agreements and objectives) and continuously monitoring it to ensure that it meets the required performance level can prevent both finger-pointing and lost time in problem determination.

For the above reasons, IBM recommends that all instances of IBM Tivoli Directory Server be benchmarked, SLA/SLOs be defined, and operational measurements be taken on a regular basis to ensure that performance objectives are being met.

© Copyright IBM Corp. 2007. All rights reserved.

1.1 Common service level agreements and objectives

Let us take a look at some general guidelines and some Lightweight Directory Access Protocol (LDAP) specific details.

1.1.1 SLA and SLO guiding principles

In order to be meaningful, service level agreements/objectives (SLA/SLOs) for a specific entity within a multi-tiered application must conform to three primary rules:

- The SLA must be measurable. The SLA must be subject to programmatic or manual measurements that provide repeated nonsubjective values.
- The SLA must measure items under the control of the application, the entity, or both. Including items that are not under the direct control of the application or entity (for example, network transition time) falls under the heading of an *end-to-end* SLA/SLO for which there should be an individual component part SLAs/SLOs.
- The SLA/SLO must be reasonable and obtainable with the design and topology chosen.

1.1.2 Lightweight Directory Access Protocol specific SLAs

For the IBM Tivoli Directory Server, all SLA/O measurements are defined as the average time taken for internal processing and the start of the data delivery. By measuring only the time to process the operation (bind, add, modify, search, and so on), we measure only the items for which the Directory Server has control. A large result set might take considerable time for a client application to retrieve, parse, and make use of.

In addition, these SLAs/SLOs are specified as an average time, not as an absolute or maximum value. A search operation that returns a result set of a thousand objects always takes longer than a single value result set.

Average bind time

This is measured as the average time in milliseconds, measured from the receipt of a correctly formatted and valid bind operation to the initial return of the success operation and data to the client application.

Average unbind time

This is measured as the average time in milliseconds, measured from the receipt of a correctly formatted and valid unbind operation to the initial return of the success operation and data to the client application.

Average add time

This is measured as the average time in milliseconds, measured from the receipt of a correctly formatted and valid add operation to the initial return of the success operation and data to the client application.

Average modify time

This is measured as the average time in milliseconds, measured from the receipt of a correctly formatted and valid modify operation to the initial return of the success operation and data to the client application.

Average delete time

This is measured as the average time in milliseconds, measured from the receipt of a correctly formatted and valid delete operation to the initial return of the success operation and data to the client application.

Average search time

This is measured as the average time in milliseconds, measured from the receipt of a correctly formatted and valid search operation to the initial return of the success operation and data to the client application. Measuring from the receipt of the valid search operation request to the initial but not final set of data to the client prevents large search results from skewing the average.

2

Does your directory have a cold? Time to do a health check

The Directory Server is a constantly changing entity and from time to time, a health check has to be done to ensure that it is working at its potential. IBM is commonly called in to check over an enterprise directory to determine why the performance appears to be slowing or not meeting the required levels. This is called a *health check* or *performance tuning engagement*.

The following section defines a list of questions about the system and data that has to be gathered to make concrete recommendations. You have to know what you have to work with before you can make the right decisions.

© Copyright IBM Corp. 2007. All rights reserved.

2.1 Questions to ask

Here is a list of questions that should be asked about each Lightweight Directory Access Protocol (LDAP) environment.

- What version of IBM Tivoli Directory Server is it?
- What fix packs have been applied to the Directory Server?
- Is there a mix of Directory Server versions on this system?
- What version of IBM DB2 is being used?
- What fix packs have been applied to IBM DB2?
- What operating system (OS) version is the Directory Server running on?
- What fix packs have been applied to the OS?
- Which applications are using this Directory Server?
- How many users are using this Directory Server?
- How many entries are in the database?
- How many processors are on the server?
- How much memory is on the server?
- Is the database on local or storage area network (SAN) drives?
- Are you running Redundant Array of Independent Disks (RAID)?
- How many master or peer-to-peer master servers are in this system?
- How many replicas are in this system?
- How many Forwarder or Gateway servers are in this system?
- Do you have a wide area network (WAN) environment and if so, how many and how fast are they?
- What is your failover process?
- What is your backup process?
- Do you have a distributive or central administration?

2.2 Diagrams and layouts of the system

In addition to asking the above questions you also have to obtain the following information:

- Physical layout of the Directory Servers
- Logical layout of the Directory Servers

2.3 Configurations, logs, and outputs

In determining the health of the directory, reviewing the logs should be the first item performed. The following section describes the log files available to you as part of the review process.

Depending on the version of the IBM Tivoli Directory Server you are using and what OS you are running, the configuration and error logs will be in different locations.

The following list provides the default locations of where the software puts them unless you make changes to the configuration files instructing the software to write to a different location.

- Version 4.x:
  - Directory Server configuration files:
    - UNIX®: /usr/ldap/etc
    - Solaris™: /opt/IBMldaps/etc
    - Windows: <Drive>:\Program Files\IBM\ldap\etc
  - Directory Server error logs:
    - UNIX: /var/ldap/
    - Windows: <Drive>:\Program Files\IBM\ldap\var\
  - IBM DB2 error logs:
    - UNIX: <instance_home_directory>/<instance_name>/sqllib/db2dump
    - Windows: <Drive>:<instance_home_directory>\<instance_name>\sqllib\db2dump
- Version 5.x:
  - Directory Server configuration files:
    - UNIX: /usr/ldap/etc
    - Solaris: /opt/IBMldaps/etc
    - Windows: <Drive>:\Program Files\IBM\ldap\etc
  - Directory Server error logs:
    - UNIX: /var/ldap/
    - Windows: <Drive>:\Program Files\IBM\ldap\var\
  - IBM DB2 error logs:
    - UNIX: <instance_home_directory>/<instance_name>/sqllib/db2dump
    - Windows: <Drive>:<instance_home_directory>\<instance_name>\sqllib\db2dump
- Version 6.x:
  - Directory Server configuration files:
    - UNIX: <instance_home_directory>/<instance_name>/idsslapd-<instance_name>/etc
    - Solaris: /opt/IBMldaps/etc
    - Windows: <Drive>:\Program Files\IBM\ldap\etc
  - Directory Sever error logs:
    - UNIX: <instance_home_directory>/<instance_name>/idsslapd-<instance_name>/logs
    - Windows: <Drive>:\Program Files\IBM\ldap\var\
  - IBM DB2 error logs:
    - UNIX: <instance_home_directory>/<instance_name>/sqllib/db2dump
    - Windows: <Drive>:<instance_home_directory>\<instance_name>\sqllib\db2dump

The following files and logs can all be used to understand what the Directory Server is doing and how it is configured and tuned. This document includes tools that will assist you in problem determination and possible ways to correct them.

- For IBM Tivoli Directory Server:
  - Version 4.x:
    - *slapd32.conf*: This is where you look for LDAP cache settings and DB2 database connections.
    - *errors.log*: This is where you look for problems with the schema and abnormal errors being posted.

        - *audit*: This log can used to obtain information about what the customer is really doing with the directory.
    - Version 5.x and later:
        - *ibmslapd.conf*: This is where you look for LDAP cache settings and DB2 database connections.
        - *ibmslapd.log*: This is where you look for problems with the schema and abnormal errors being posted.
        - *db2cli.log*: This log shows any communication errors between the Directory Server and DB2 database.
        - *audit.log*: This log can be used to obtain information about what the customer is really doing with the directory. We discuss this log in more detail later in this document.
    - Version 6.x and later:
        - *lostandfound.log*: This log shows replication conflicts, if any. If there are lots of conflicts being logged, this affects the performance of the server.
- For IBM DB2:
    - Version 4x and later:
        - *db2diag.log*: This log tells you of any problems with the database and shows when someone kicks off a reorg or other action against the database. Each entry is time stamped.
        - *idsldap.nfy*: This log is like a short version of the db2diag.log and shows just the problems with the instance without lots of operating details.
        - *cli.log (4x only) or db2cli.log (5x and later)*: This log shows any communication errors between the Directory Server and DB2 database.

2.4 Scripts to help gather information

We have built scripts that can be run to gather most of the required configuration settings for both the LDAP and DB2. These scripts help you to identify the current state of the Directory Server.

Note: Refer to Appendix G, “Additional material” on page 193, for details about how to obtain the additional files mentioned throughout this IBM Redpaper.

2.4.1 perfcheck_database.sh

This script gathers configuration information about IBM DB2 and is run as the instance owner of IBM DB2. This script gathers the following information:

- DB2 environmental settings
- DB2 database manager configuration settings
- DB2 instance configuration settings
- DB2 buffer pool settings
- List of the tablespaces used in this instance

It also prints out a db2look output that shows the structure of the database instance along with how each of the tables and indexes is built.

1. sudo (or su) to the DB2 instance owner and then source the DB2 environment variables by db2profile (if this is not performed, the db2 commands will not work). The script file db2profile is located in the sqllib subdirectory under the instance owner's home directory. If you have to tailor this file, follow the comments inside the file to set your instance name, user paths, and default database name (the default path is /home/ldapdb2/sqllib/db2profile.) Then run the script. If, for example, your instance owner is ldapdb2:

   ```
   #su - ldapdb2
   #.  /home/ldapdb2/sqllib/db2profile
   ```

2. Change to the directory where your ldapscripts are located:

   ```
   #cp /opt/tmp/ldapscripts
   # ./perfcheck_database.sh ldapdb2 > /tmp/perfcheck_database.out 2>&1
   ```

3. When done, type `exit` to return to root ID.

2.4.2 perfcheck_runstats.sh

This gathers information about when the last time **`runstats`** was run on the DB2 tables and indexes used in the LDAP instance. This is run as the instance owner of DB2.

This script gathers information about each of the tables and indexes, which are listed with the date and time the last **`runstats`** was run.

1. sudo (or su) to the DB2 instance owner and then source the DB2 environment variables by db2profile (if this is not performed, the db2 commands will not work). The script file db2profile is located in the sqllib subdirectory under the instance owner's home directory. If you have to tailor this file, follow the comments inside the file to set your instance name, user paths, and default database name (the default path is /home/ldapdb2/sqllib/db2profile.) Then run the script. If, for example, your instance owner is ldapdb2:

   ```
   #su - ldapdb2
   #.  /home/ldapdb2/sqllib/db2profile
   ```

2. Change to the directory where your ldapscripts are located:

   ```
   #cp /opt/tmp/ldapscripts
   # ./perfcheck_runstats.sh ldapdb2 > /tmp/perfcheck_runstats.out 2>&1
   ```

3. When done, type `exit` to return to root ID.

2.4.3 perfcheck_system.sh

This script gathers information about the OS that the Directory Server is running on. This is run as the root user of the OS.

This script gathers the following information depending on the type of UNIX OS:

- Memory
- SWAP space
- How many and type of CPUs
- Kernel level
- Outputs for:
  - vmstat
  - iostat

- Solaris only
  - prstat output
- Copy of /etc/system file for OS settings

For example:

```
# ./perfcheck_system.sh >/tmp/perfcheck_system.out 2>&1
```

2.4.4 perfcheck_ldap.sh

This script gathers information about the Directory Server configuration. This is run as the root user.

This script gathers the following information:

- The following information is gathered from the LDAP configuration file:
  - How many DB2 connections
  - What the LDAP cache settings are set to
- Then it prints out the LDAP configuration file (ibmslapd.conf).
- Outputs for the following **`ldapsearch`** commands:
  - **`ldapsearch -b "" -s base objectclass=*`**

    This search tells you what suffixes are created, what version of LDAP is running, what type of LDAP server this is (master, replica, and so on), and the port number being used. Depending on the version of LDAP, it also shows you if you are in configuration mode or not.
  - **`ldapsearch -b cn=monitor -s base "objectclass=*"`**

    This search, depending on the version of LDAP, gives you a snapshot of the directory statistics from the last time it was started to the point of the snapshot. These stats get reset when you restart the LDAP.

Make sure that you have the path variable to the LDAP configuration file correct or you will get some errors about some of the commands in this script. For example, for 4.x and 5.x (most of the time), it is /usr/ldap/etc, but for 6.0, it is in the home directory of the LDAP instance because you can have more than one LDAP instance on a server.

If you are running a 4.x server or earlier, you will receive an error saying that you cannot find the ibmslapd.conf file; this is expected. If you are running a 5.x server or later, you will receive an error saying that you cannot find the slapd32.conf file; this is expected.

For example, if you are running 6.0 and your DB2 instance owner home directory is called ldapdb2:

```
# ./perfcheck_ldap.sh "/home/ldapdb2/idsslapd-ldapdb2/etc"
>/tmp/perfcheck_ldap.out 2>&1
```

2.5 IBM DB2 monitors

The next piece of information that helps you to determine what tuning, if any, the LDAP database needs, requires you to turn on special DB2 monitors to gather timings, stats, and list types of SQL searches being run against the database.

Collecting system monitor data introduces processing overhead for the database manager. For example, in order to calculate the execution time of SQL statements, the database manager must make calls to the operating system to obtain timestamps before and after the execution of every statement. These types of system calls are generally expensive. Another form of overhead incurred by the system monitor is increased memory consumption. For every monitor element tracked by the system monitor, the database manager uses its memory to store the collected data.

Care has to be taken with turning on the *DFT_MON_TABLE monitor switch* as this will be a performance hit on the database. This switch should only be enabled when needed and then disabled.

The following monitors shown in Table 2-1 can be turned on to gather stats. By default, all switches are turned off, except DFT_MON_TIMESTAMP.

Table 2-1 DB2 monitor overview

Monitor switch	DBM parameter	Information provided
BUFFERPOOL	DFT_MON_BUFPOOL	Number of reads and writes, time taken of all the buffer pools
LOCK	DFT_MON_LOCK	Lock wait times, deadlocks
SORT	DFT_MON_SORT	Number of heaps used, sort performance
STATEMENT	DFT_MON_STMT	Start/stop time, statement identification
TABLE	DFT_MON_TABLE	Measure of activity (rows read/written)
UOW	DFT_MON_UOW	Start/end times, completion status of Unit of Work
TIMESTAMP	DFT_MON_TIMESTAMP	Timestamps

We have provided scripts to help you turn on and turn off these monitor switches. Each time you turn the monitor switches on or off, you have to stop and start DB2.

- perftune_enablemonitor.sh: Turn on all monitors except DFT_MON_TABLE
- perftune_enablemonitor_all.sh: Turns on all monitors
- perftune_disablemonitor.sh: Turns off all monitors except DFT_MON_TIMESTAMP

1. Stop the Directory Server. For example, for IBM Tivoli Directory Server 6.0 with an instance name of ldapdb2:

   ```
   idsslapd -I ldapdb2 -k
   ```

2. Now sudo (or su) to the DB2 instance owner and then source the DB2 environment variables with db2profile (if this is not performed, db2 commands will not work). The script file db2profile is located in the sqllib subdirectory under the instance owner's home directory. If you have to tailor this file, follow the comments inside the file to set your instance name, user paths, and default database name (the default path is /home/ldapdb2/sqllib/db2profile.) Then run the script. For example, if your instance owner is ldapdb2:

   ```
   #su - ldapdb2
   #.  /home/ldapdb2/sqllib/db2profile
   ```

3. Change to the directory where your ldapscripts are located. Run one of the three following monitor scripts to either turn on or turn off the monitors.

   ```
   ./perftune_enablemonitor.sh
   ```

   Or:

   ```
   ./perftune_enablemonitor_all.sh
   ```

   Or:

   ```
   ./perftune_disablemonitor.sh
   db2stop
   db2start
   ```

4. Type `exit` to return to the root ID.
5. Start the Directory Server back up.

   ```
   idsslapd -I ldapdb2
   ```

2.6 Analyzing the gathered information

The first step is to look over the outputs from the perfcheck scripts and then apply the tools in the next section to analyze and make recommendations to improve the throughput of directory.

There are specific settings that improve performance when used with large enterprise environments with hundreds of millions of entries. For example, with 1 million entries in a directory, the Directory Server cache does not increase performance and in fact can degrade performance to filter cache invalidation. In these instances, it is better to allocate the memory to DB2 and bypass the Directory Server caches with smaller directories. What we suggest that you do is to drop the filter cache setting down to 100 and use the performance tools included in this book to determine what the other LDAP cache settings should be.

Run the monitor script for a few days to see what types of loads are being used, how much memory and how much of the Directory Server caches are being used.

The important thing to remember when tuning the IBM Tivoli Directory Server is that it is a continual process. The interaction between the transaction mix, the Directory Server process, and DB2 has to be evaluated and tuned on a regular basis to ensure the best performance possible from the directory. Directories change from time to time as data is added, changed, or deleted. More applications are using LDAP not just for their authentication needs but also as their authorization database. Therefore, the need to do benchmarking of the directory is very important. With benchmarking, you have a place to start from and compare back to. The benchmarks enable you to identify when your system is not running up to par. When identified, further testing is required to improve the outcome.

3

Tools to help you assist with your DB2 tuning

This chapter introduces a Microsoft® Excel® spreadsheet and a script that can be used for DB2 tuning.

© Copyright IBM Corp. 2007. All rights reserved.

3.1 DB2-Config-calc-tool-template Excel sheet

This Excel sheet was designed to work with DB2 8.1 with Fix Packs 9 through 13 (to date) to help set up the db2_tunings.sh file. The spreadsheet requires the following information, each of which is covered in a later section:

- Turning on the DB2 monitor switches (use perftune_enablemonitor.sh, as described in 6.3, “perftune_enablemonitor_all.sh” on page 48) and run the monitors during the high loads of the database, preferably for a 24-hour period.
- Run the perfanalyze_getallsnapshots.sh (included in the IBM Redpaper download) to get the snapshot-database.<date.time> output.
- Retrieve the DB2 instance configuration. This information can be retrieved by the perfcheck_database.sh script (see 2.4.1, “perfcheck_database.sh” on page 8).

Steps to use the spreadsheet:

1. Open the DB2-Config-calc-tool-template Excel sheet and select the **DBCFG** tab at the bottom.
2. Open the perfcheck_database.out file (this is the output from perfcheck_database.sh) and scroll down to the line called *Database Configuration for Database <instance name>* and copy to the clipboard from the next line up to and including *Automatic reorganization*, as shown in Example 3-1.

Example 3-1 Database configuration for database

```
Database Configuration for Database ldapdb2
Database configuration release level                    = 0x0a00   <= start copying at this line
Database release level                                  = 0x0a00
Database territory                                      = US
Database code page                                      = 1208
Database code set                                       = UTF-8
Database country/region code                            = 1
Database collating sequence                             = BINARY
Alternate collating sequence             (ALT_COLLATE) =
Database page size                                      = 4096
...
Automatic maintenance                     (AUTO_MAINT) = OFF
Automatic database backup           (AUTO_DB_BACKUP) = OFF
Automatic table maintenance         (AUTO_TBL_MAINT) = OFF
Automatic runstats                 (AUTO_RUNSTATS) = OFF
Automatic statistics profiling   (AUTO_STATS_PROF) = OFF
Automatic profile updates        (AUTO_PROF_UPD) = OFF
Automatic reorganization              (AUTO_REORG) = OFF   <= end copying this line
```

3. Open a new Microsoft Word document and paste what you just copied.
4. Convert database configuration from table.
   a. Click **Edit** → **Select All**.
   b. Click **Table** → **Convert** → **Text to Table**.
5. The window shown in Figure 3-1 is displayed. In this widow, perform the following steps:
   a. Select **2** from the Number of columns drop-down menu.
   b. In the AutoFit behavior section, select **Fixed column with** and select **Auto**.
   c. In the “Separate text at” section, select **Other** at and enter an equals to sign (=) in the box.
   d. Click **OK**.

This converts the highlighted text to a table.

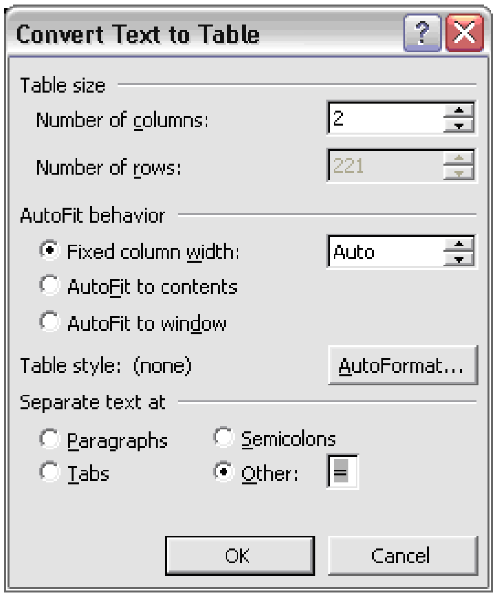


Figure 3-1 Convert text to table

6. Cut the whole area of the converted table using “Ctrl+x” and go back to the Excel sheet. From the DBCFG tab, click the “B4” cell and paste the converted table using “Ctrl+v”. This replaces the data on this sheet with the new converted data.
7. Open the snapshot-database. <date.time> output file and scroll down to the line called “Database Snapshot”. Copy to the clipboard, starting from the next line, and up to and including “Configured size (bytes)” line as shown in Example 3-2.

Example 3-2 Snapshot-database excerpt (continued)

```
Database Snapshot
Database name                              = LDAPDB2 <= start copying at this line
Database path                              = /opt/export/home/ldapdb2/ldapdb2/NODE0000/SQL00001/
Input database alias                       = LDAPDB2
Database status                            = Active
Catalog database partition number          = 0
Catalog network node name                  =
Operating system running at database server= SUN
Location of the database                   = Local
First database connect timestamp           = 05/02/2006 13:35:29.418903
Last reset timestamp                       =
Last backup timestamp                      = 04/26/2006 15:08:37.000000
Snapshot timestamp                         = 05/03/2006 14:56:45.602867
...
Memory Pool Type                           = Other Memory
Current size (bytes)                    = 0
High water mark (bytes)                 = 0
Configured size (bytes)                 = 12353536      <= end copying this line
```

8. Open a new Microsoft Word document and paste what you just copied.
9. Convert database configuration from table.
   a. Click **Edit** → **Select All**.
   b. Click **Table** → **Convert** → **Text to Table**.

10. The same window as shown in Figure 3-1 on page 15 is displayed. In this window, perform the following steps:

    a. Select **2** from the Number of columns drop-down list.
    b. In the AutoFit behavior section, select **Fixed column with** and select **Auto**.
    c. In the “Separate text at” section, select **Other** and enter an equals to sign (=) in the box.
    d. Click **OK**.

    This converts the highlighted text to a table.

11. Cut this whole area of the converted table using “Ctrl+x” and go back to the Excel sheet. From the Snapshot tab, click the “B4” cell and paste the converted table using “Ctrl+v”. This replaces the data on this sheet with the new converted data.

12. Click the **Results** tab of the DB2-Config-calc-tool-template Excel sheet.

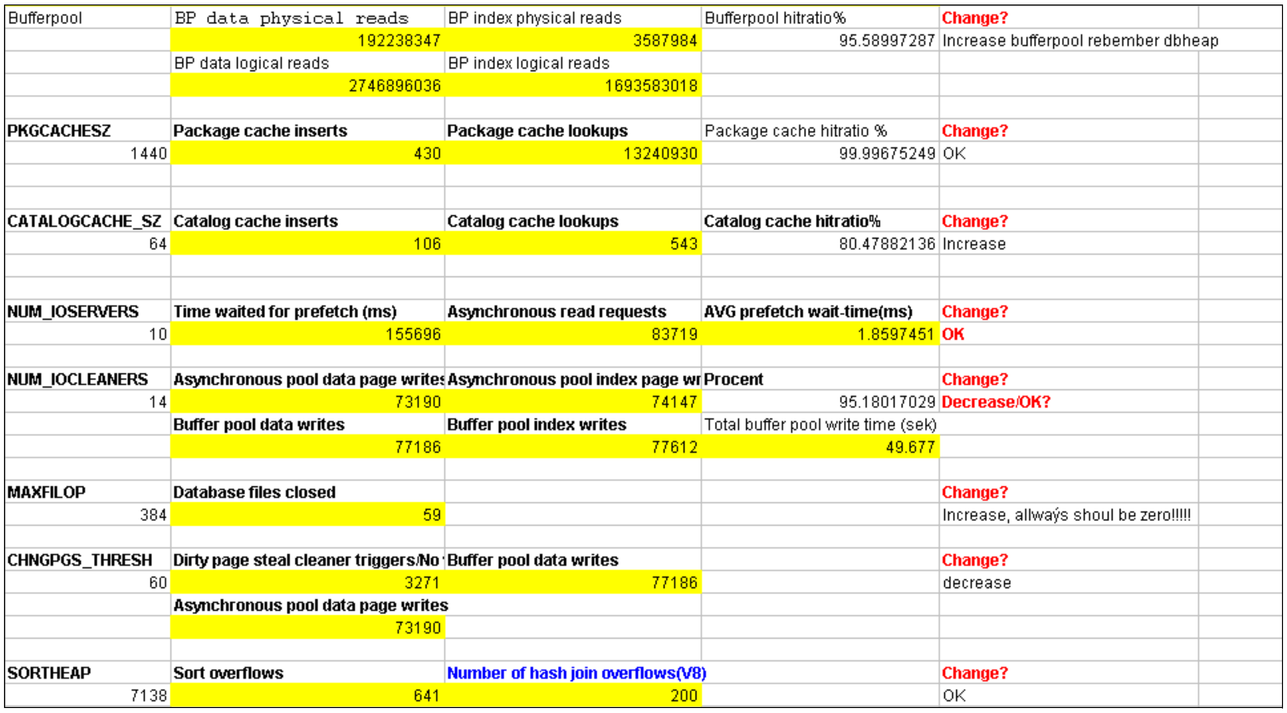
    This shows you the results from what the LDAP/DB2 was configured at this time compared to what performance results that were logged for this DB2 during the capture.

    This is an example of using a part of the output. As you can see, it tells you that you have to increase or decrease a setting.

13. When you get the output, you have to edit the db2_tunings.sh (see 3.2, “db2_tunings.sh” on page 17) and plug in new settings for those configurations that require updating.

14. Apply the new settings to DB2, restart, and test and reset the monitors. Monitor the LDAP for another 24 hours and pull a new set of monitor outputs and rerun through the Excel sheet to check if the settings are okay.

    Figure 3-2 is an example of what the spreadsheet can look like when you input the above information.



Bufferpool	BP data physical reads	BP index physical reads	Bufferpool hitratio%	Change?	
	192238347	3587984	95.58997287	Increase bufferpool rebember dbheap	
	BP data logical reads	BP index logical reads			
	2746896036	1693583018			
PKGCACHESZ	Package cache inserts	Package cache lookups	Package cache hitratio %	Change?	
1440	430	13240930	99.99675249	OK	
CATALOGCACHE_SZ	Catalog cache inserts	Catalog cache lookups	Catalog cache hitratio%	Change?	
64	106	543	80.47882136	Increase	
NUM_IOSERVERS	Time waited for prefetch (ms)	Asynchronous read requests	AVG prefetch wait-time(ms)	Change?	
10	155696	83719	1.8597451	OK	
NUM_IOCLEANERS	Asynchronous pool data page writes	Asynchronous pool index page wr	Procent	Change?	
14	73190	74147	95.18017029	Decrease/OK?	
	Buffer pool data writes	Buffer pool index writes	Total buffer pool write time (sek)		
	77186	77612	49.677		
MAXFILOP	Database files closed			Change?	
384	59			Increase, allwaýs shoul be zero!!!!!	
CHNGPGS_THRESH	Dirty page steal cleaner triggers/No	Buffer pool data writes		Change?	
60	3271	77186		decrease	
	Asynchronous pool data page writes				
	73190				
SORTHEAP	Sort overflows	Number of hash join overflows(V8)		Change?	
7138	641	200		OK	

Figure 3-2 Spreadsheet tuning results

3.2 db2_tunings.sh

Out of the box, the Directory Server database is tuned for a small directory and must be tuned to run correctly for medium or large directories. For example, the DB2 buffer pools are set by default to use only 256 MB of memory. With the performance improvements with DB2 8.2 and later (8.1 with Fix Pack 9 or later) and the many changes that have been made with IBM Directory Server V6.0 through fix packs, there are a number of improvements to the performance of the Directory Server. Some big improvements deal with how we index attributes and how we process searches. Another improvement or change is the amount of memory we recommend to the DB2 buffers; with large directories we find that the *best throughput* comes with *larger* DB2 buffers. This script starts off with figuring out how much real memory you have on this server and recommend taking half of it for the DB2 buffers. This can be raised or lowered depending on the requirement. Use this information along with the output of the DB2-Config-calc-tool-template to help set up the many settings in this script.

The following settings can be changed in this script. These setting are pre-set up for over 1 million entries and can be changed up or down as required by using the DB2-Config-calc-tool-template after running your benchmarks with the DB2 monitors. In Chapter 4, "DB2 settings related to LDAP" on page 23, we cover each of these settings.

- SHEAPTHRES=30000
- DBHEAP=2000
- CATALOGCACHE_SZ=64
- CHNGPGS_THRESH=60
- SORTHEAP=7138
- MAXLOCKS=80
- LOCKTIMEOUT=120
- LOCKLIST=400
- MINCOMMIT=1
- UTIL_HEAP_SZ=5000
- APPLHEAPSZ=2048
- STAT_HEAP_SZ=5120
- DFT_PREFETCH_SZ=32
- MAXFILOP=384
- MAXAPPLS=100
- PCKCACHESZ=1440
- DFT_EXTENT_SZ=32
- LOGFILSIZ=5000
- LOGPRIMARY=5
- LOGSECOND=60

3.2.1 Using the script

The db2_tunings.sh script was designed to be used to pre-tune the database.

```
$ /opt/tmp/tools/tuning/db2_tunings.sh -?
```

- Usage:

  ```
  db2_tunings.sh -s size [-r ratio] [-e] [-db dbname]
  db2_tunings.sh -ibp ibmdefaultbp -lbp ldapbp [-db dbname]
  ```

- Options:

db dbname	DB name to update (Default=ldapdb2)
s size	Target memory usage in MB
r ratio	Buffer pool ratio, IBMDefaultBP/LdapBP (Default=3)

ibp ibmdefbp	Size of ibmdefaultbp buffer pool setting
lbp ldapbp	Size of ldapbp buffer pool setting
e	Estimate buffer pools only. If size is not specified, the amount of system memory is determined and SIZE is set to 50% of the total. If system memory cannot be determined, size must be specified.

Note that this script must be executed as the DB2 instance owner.

Estimate the memory usage for the buffer pools. The -e option provides the recommended memory usage in MB.

```
$ /opt/tmp/tools/tuning/db2_tunings.sh -e
```

The output is similar to the following:

```
Memory Detected: 4096
Recommended Memory Size: 2048.00 MB (assumes 50% of total memory)
Recommended ibmdefaultbp setting: 393216 (assumes RATIO=3)
Recommended ldapbp setting: 16384 (assumes RATIO=3)
```

Tune the database using the required memory usage size (in MB). The recommendation of 50% of total memory provided using the -e option is only a guideline. From the example above, <size> would be 2048.

Note: The optimum memory size to allocate for the database depends both on the amount of physical memory and on the operating system. Buffer pools are used for memory mapped I/O, therefore they should not be configured to swap. Physical memory, not virtual memory, is used for buffer pools. Tivoli Directory Server 5.2 and 6.0 versions are implemented using 32-bit code and calling a 32-bit version of DB2, on all platforms except AIX. As a result, for all platforms except AIX, the maximum amount of memory addressable by the DB2 process is 4 GB, and this includes both code and data. In practice, only about 2.5 GB can be used for data, including the buffer pools.

If more than one instance of Tivoli Directory Server is to be run on a single server, the available physical memory must be divided between instances. You have to reduce the -e option accordingly, so as not to allocate more memory than you really have.

```
$ /opt/tmp/tools/tuning/db2_tunings.sh -s 2048 -db ldapdb2
```

This script can and should be edited to fit the requirements of your database. The settings that we used in this script were a starting point for a directory that had over 1 million entries. When you edit this script, the only places you should have to change, if needed, will be the ones between the constants remarks as shown in Example 3-3.

Example 3-3 Changeable parameters for the db2_tunings.sh script

```
###############
## CONSTANTS ##
###############

SHEAPTHRES=30000
DBHEAP=3000
CATALOGCACHE_SZ=64
CHNGPGS_THRESH=60
SORTHEAP=7138
MAXLOCKS=80
LOCKTIMEOUT=120
```

```
LOCKLIST=400
MINCOMMIT=1
UTIL_HEAP_SZ=5000
APPLHEAPSZ=2048
STAT_HEAP_SZ=5120
## IOCLEANERS and IOSERVERS are now calculated based on the ldapbp value
#NUM_IOCLEANERS=8
#NUM_IOSERVERS=6
DFT_PREFETCH_SZ=32
MAXFILOP=384
MAXAPPLS=100
PCKCACHESZ=1440
# If you are going to change the DFT_ENTENT_SZ you will also need to
#    remove the comment character from the "db2 update" command later on in
#    this script.    HINT: Search for EXTENT to find the command.
DFT_EXTENT_SZ=32

LOGFILSIZ=5000
LOGPRIMARY=5
LOGSECOND=60

# If you are going to change the NEWLOGPATH you will also need to
#    remove the comment character from the "db2 update" command later on in
#    this script.    HINT: Search for NEWLOGPATH to find the command.
NEWLOGPATH=/usr/logfiles/logs

######################
## END DB CONSTANTS ##
######################
```

From your findings with the DB2-Config-calc-tool-template and other tools used in this document, you can come up with the required settings for this script.

The output is similar to the one shown in Example 3-4.

Example 3-4   db2_tunings.sh output

```
Database Connection Information
Database server       = DB2/SUN 8.2.3
SQL authorization ID   = LDAPDB2
Local database alias   = LDAPDB2
The current buffer pool settings are as follows:
BPNAME                          NPAGES    PAGESIZE
IBMDEFAULTBP                    29500     4096
LDAPBP                          1230      32768
2 record(s) selected.
Updating the buffer pool settings.

SQL20189W The buffer pool operation (CREATE/ALTER) will not take effect until the
next database startup due to insufficient memory. SQLSTATE=01657

SQL20189W The buffer pool operation (CREATE/ALTER) will not take effect until the
next database startup due to insufficient memory. SQLSTATE=01657

The DB2 configuration parameters settings are as follows:
```

```
Database heap (4 KB)                    (DBHEAP) = 1200
Catalog cache size (4 KB)               (CATALOGCACHE_SZ) = (MAXAPPLS*4)
Utilities heap size (4 KB)              (UTIL_HEAP_SZ) = 5000
Buffer pool size (pages)                (BUFFPAGE) = 1000
Max storage for lock list (4 KB)        (LOCKLIST) = 100
Sort list heap (4 KB)                   (SORTHEAP) = 256
Default application heap (4 KB)         (APPLHEAPSZ) = 2048
Package cache size (4 KB)               (PCKCACHESZ) = 360
Statistics heap size (4 KB)             (STAT_HEAP_SZ) = 4384
Percent. of lock lists per application  (MAXLOCKS) = 10
Number of asynchronous page cleaners    (NUM_IOCLEANERS) = 1
Number of I/O servers                   (NUM_IOSERVERS) = 3
Default prefetch size (pages)           (DFT_PREFETCH_SZ) = AUTOMATIC
Default tablespace extentsize (pages)   (DFT_EXTENT_SZ) = 32
Max number of active applications       (MAXAPPLS) = AUTOMATIC
Max DB files open per application       (MAXFILOP) = 64
Group commit count                      (MINCOMMIT) = 1

Updating the DB2 config settings

DB20000I The UPDATE DATABASE MANAGER CONFIGURATION command completed successfully.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.

SQL1363W One or more of the parameters submitted for immediate modification were
not changed dynamically. For these configuration parameters, all applications must
disconnect from this database before the changes become effective.

DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.

SQL1363W One or more of the parameters submitted for immediate modification were
not changed dynamically. For these configuration parameters, all applications must
disconnect from this database before the changes become effective.

DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.

SQL1363W One or more of the parameters submitted for immediate modification were
not changed dynamically. For these configuration parameters, all applications must
disconnect from this database before the changes become effective.

DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.

SQL1363W One or more of the parameters submitted for immediate modification were
not changed dynamically. For these configuration parameters, all applications must
disconnect from this database before the changes become effective.

DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
```

```
SQL1363W One or more of the parameters submitted for immediate modification were
not changed dynamically. For these configuration parameters, all applications must
disconnect from this database before the changes become effective.

DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.

The DB2 configuration parameters settings are as follows:

Database heap (4 KB)                    (DBHEAP) = 2000
Catalog cache size (4 KB)               (CATALOGCACHE_SZ) = (MAXAPPLS*4)
Utilities heap size (4 KB)              (UTIL_HEAP_SZ) = 5000
Buffer pool size (pages)                (BUFFPAGE) = 1000
Max storage for lock list (4 KB)        (LOCKLIST) = 1000
Sort list heap (4 KB)                   (SORTHEAP) = 7138
Default application heap (4 KB)         (APPLHEAPSZ) = 2048
Package cache size (4 KB)               (PCKCACHESZ) = 1440
Statistics heap size (4 KB)             (STAT_HEAP_SZ) = 5120
Percent of lock lists per application   (MAXLOCKS) = 80
Number of asynchronous page cleaners    (NUM_IOCLEANERS) = 18
Number of I/O servers                   (NUM_IOSERVERS) = 16
Default prefetch size (pages)           (DFT_PREFETCH_SZ) = AUTOMATIC
Default tablespace extentsize (pages)   (DFT_EXTENT_SZ) = 32
Max number of active applications       (MAXAPPLS) = AUTOMATIC
Max DB files open per application       (MAXFILOP) = 384
Group commit count                      (MINCOMMIT) = 1

DB20000I The TERMINATE command completed successfully.
DB20000I The FORCE APPLICATION command completed successfully.
DB21024I This command is asynchronous and may not be effective immediately.
03/08/2006 16:34:47     0   0   SQL1064N DB2STOP processing was successful.
SQL1064N DB2STOP processing was successful.
03/08/2006 16:34:49     0   0   SQL1063N DB2START processing was successful.
SQL1063N DB2START processing was successful.
Database Connection Information
Database server        = DB2/SUN 8.2.3
SQL authorization ID   = LDAPDB2
Local database alias   = LDAPDB2
The new buffer pool settings are as follows:
BPNAME                              NPAGES       PAGESIZE
IBMDEFAULTBP                        393216       4096
LDAPBP                              16384        32768
2 record(s) selected.
DB20000I The TERMINATE command completed successfully.

Defining the transaction log size parameters allow for the worst case of ACL
propagation. The chosen setting will allow ACLs to propagate from a suffix to up
to 3 million Access Manager users. This setting can use up to 1.2 GB additional
disk space in the DB2 instance owner home directory.
```

```
The number of log file size will be increased to 5000.
The number of primary log buffers will be increased to 5.
The number of secondary log buffers will be increased to 60.

Adjust the LOGSECOND setting for more or less users. Ensure the disk space is
available for whatever setting is used, since running out of disk space for the
transaction log can corrupt the database and require reloading of the database.

The default transaction log settings are as follows:
Log file size (4 KB)                         (LOGFILSIZ) = 2000
Number of primary log files                (LOGPRIMARY) = 3
Number of secondary log files               (LOGSECOND) = 2
Changed path to log files                  (NEWLOGPATH) =

Path to log files =/opt/export/home/ldapdb2/ldapdb2/NODE0000/SQL00001/SQLOGDIR/

Updating the transaction log settings.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
03/08/2006 16:35:02     0   0   SQL1064N DB2STOP processing was successful.
SQL1064N DB2STOP processing was successful.
03/08/2006 16:35:04     0   0   SQL1063N DB2START processing was successful.
SQL1063N DB2START processing was successful.

The new transaction log settings are as follows:
Log file size (4 KB)                         (LOGFILSIZ) = 5000
Number of primary log files                (LOGPRIMARY) = 5
Number of secondary log files               (LOGSECOND) = 60
Changed path to log files                  (NEWLOGPATH) =

Path to log files = /opt/export/home/ldap
db2/ldapdb2/NODE0000/SQL00001/SQLOGDIR/

Verifying that the file system for the transaction logs is large enough to
accommodate the maximum growth.

logfilsiz=5000 logprimary=5 logsecond=60 logspace=1300000
The available file space on
/opt/export/home/ldapdb2/ldapdb2/NODE0000/SQL00001/SQLOGDIR/ is 41812992 KB.
The log parameters allow the log files to grow to a maximum of 1300000 KB.
Check succeeded: There is sufficient disk space to allow for the maximum log file
growth.
```

4

DB2 settings related to LDAP

All the settings covered in this chapter can be changed in the db2_tunings.sh script (see 3.2, “db2_tunings.sh” on page 17). Using the DB2-Config-calc-tool-template Excel sheet results tab along with this section, you can have a better understanding of appropriate values for these settings.

© Copyright IBM Corp. 2007. All rights reserved.

4.1 SHEAPTHRES: Sort heap threshold

The *Sort Heap Threshold* parameter, as a database manager configuration parameter, applies across the entire DB2 instances. This threshold prevents the database manager from using excessive amounts of memory for large number of sorts for any and all instances that this DB2 will use.

This is the only database manager (DBM) setting that Lightweight Directory Access Protocol (LDAP) cares about. For most production accounts, you will only have one DB2 instance per server. With IBM Directory Server 6.0, you can have more than one LDAP instance on a server and that is where this setting will really have to be watched, because you will have to take into consideration each of the DB2 instances SORTHEAP settings for each of the LDAP instances that are on this one server to find what this setting should be.

Recommendation: Ideally, you should set this parameter to a reasonable multiple of the largest SORTHEAP parameter you have in your database manager instance. This parameter should be at least twice the largest SORTHEAP defined for any database within the instance. For LDAP, depending on the types of searches, we found that setting this to 300,000 works for most large directories. Monitor and adjust this, if needed, using the tools in this document.

4.2 DBHEAP: Database heap

There is one *database heap* per database, and the database manager uses it on behalf of all applications connected to the database. It contains control block information for tables, indexes, tablespaces, and buffer pools. It also contains space for the log buffer (logbufsz) and temporary memory used by utilities.

Recommendation: For LDAP, depending on the types of searches, we found that setting this to 2000 works depending on the types of loads you have. You might have to raise this to 3000 for some large directories. You have to monitor and adjust this, if needed, using the tools in this document.

4.3 CATALOGCACHE_SZ: Catalog cache size

The *catalog cache size* parameter is allocated out of the database shared memory, and is used to cache system catalog information. Caching catalog information allows the database manager to reduce its internal overhead by eliminating the need to access the system catalogs to obtain information that has previously been retrieved.

Recommendation: For LDAP, setting this to 64 works for most large directories. Monitor and adjust this, if needed, using the tools in this document.

4.4 SORTHEAP: Sort heap size

The *sort heap size* parameter defines the maximum number of private memory pages to be used for private sorts or the maximum number of shared memory pages to be used for shared sorts.

Recommendation: For LDAP with large directories and high traffic, the default value of 256 is too low and has to be changed. We found that setting this to 5120 works. You have to monitor and adjust this higher, if needed, using the tools in this document. Depending on your types of searches and the complexity of the search, this has to be set higher. When doing so, it is best to add in 1024 chunks.

4.5 MAXLOCKS: Maximum percentage of lock list before escalation

Lock escalation is the process of replacing row locks with table locks, reducing the number of locks in the list. This parameter defines a percentage of the lock list held by an application that must be filled before the database manager performs escalation. When the number of locks held by any one application reaches this percentage of the total lock list size, lock escalation occurs for the locks held by that application. Lock escalation also occurs if the lock list runs out of space.

Recommendation: When using the provided version of DB2, there should not be any other application running against the database other than the LDAP process (the provided version is only licensed for LDAP use). With that we found that setting this to 80% is the best for all sizes of the directory. The default of 10% is way too low.

4.6 LOCKTIMEOUT: Lock timeout

This database parameter specifies the number of seconds that an application will wait to obtain a lock. This helps to avoid global deadlocks for applications.

If you set this parameter to 0, locks are not waited for. In this situation, if no lock is available at the time of the request, the application immediately receives a -911.

If you set this parameter to -1, lock timeout detection is turned off. In this situation, a lock will be waited for (if one is not available at the time of the request) until either of the following events:

- The lock is granted.
- A deadlock occurs.

Recommendation: The default of -1 is not recommended for Tivoli Directory Server. We found that setting this to 120 fits most LDAP applications. Monitor and adjust this, if needed, using the tools in this document.

The value should be set to quickly detect waits that are occurring because of an abnormal situation, such as a transaction that is stalled (possibly as a result of a user leaving their workstation). You should set it high enough so that valid lock requests do not time out because of peak workloads, during which time, there is more waiting for locks.

4.7 LOCKLIST: Maximum storage for lock list

This database parameter indicates the amount of storage that is allocated to the *lock list*. There is one lock list per database and it contains the locks held by all applications concurrently connected to the database.

Recommendation: The default of -1 is not recommended for Tivoli Directory Server. We found that setting this to 400 fits most large LDAP applications. Monitor and adjust this, if needed, using the tools in this document. The default of 100 tends to be too low to handle the amount of locks that are going on in large LDAP directories.

4.8 MINCOMMIT: Number of commits to group

This parameter allows you to delay the writing of log records to disk, until a minimum number of commits have been performed.

Recommendation: For LDAP, this *must* be set to 1 due to replication and making sure that the data has been committed. This way you will *not* lose any data, if the LDAP stops. If this is set higher than 1, it will affect the replication and cause out-of-sync conditions.

4.9 UTIL_HEAP_SZ: Utility heap size

This database parameter indicates the maximum amount of memory that can be used simultaneously by the BACKUP, RESTORE, and LOAD (including load recovery) utilities.

Recommendation: For Tivoli Directory Server, we found that the default value of 5000 works with no issues. If you have issues with running these utilities, such as running out of space, then you should increase this value. If the parameter is set too low and no more memory is available in the overflow area, you might not be able to concurrently run the utilities.

4.10 APPLHEAPSZ: Application heap size

This database parameter defines the number of private memory pages available to be used by the database manager on behalf of a specific agent or subagent.

Recommendation: For LDAP, we found that the default value of 256 is too low and we increased this to 2048 to start with. Adjust this, if needed, using the tools in this document.

4.11 STAT_HEAP_SZ: Statistics heap size

This database parameter indicates the maximum size of the heap used in collecting statistics using the RUNSTATS command. If you find that `runstats` is taking too long to run, you will have to raise this number.

Recommendation: For LDAP, we found that the default value of 4384 is too low for large LDAP directories and we increased this to 5120 to start with. Adjust this, if needed, using the tools in this document.

4.12 CHNGPGS_THREASH: Changed pages threshold

Asynchronous page cleaners write changed pages from the buffer pool (or the buffer pools) to disk before the space in the buffer pool is required by a database agent. As a result, database agents should not have to wait for changed pages to be written out so that they might use the space in the buffer pool. This improves the overall performance of the database applications.

Recommendation: For LDAP with large directories and high traffic, we found that you have to set this to 50 at times but start with the default of 60. Monitor and adjust this, if needed, using the tools in this document.

4.13 NUM_IOCLEANERS: Number of async page cleaners

This database parameter allows you to specify the number of asynchronous page cleaners for a database. These page cleaners write changed pages from the buffer pool to disk before the space in the buffer pool is required by a database agent. As a result, database agents should not have to wait for changed pages to be written out so that they might use the space in the buffer pool. This improves the overall performance of the database applications. If the applications for a database primarily consist of transactions that update data, an increase in the number of cleaners will speed up the performance. Increasing the page cleaners also decreases the recovery time from soft failures such as power outages, because the contents of the database on disk is more up-to-date at any given time.

Recommendation: We took into consideration the following factors when we set the NUM_IOCLEANERS and NUM_IOSERVERS values with the db2_tunings.sh.

- Application type
- Workload
- Buffer pool sizes

We found through testing that if we calculate NUM_IOCLEANERS and NUM_IOSERVERS based on the buffer pool values, we then set NUM_IOSERVERS to be approximately 1 for every 1500 of LDAPBP, and then set NUM_IOCLEANERS to 2 higher than NUM_IOSERVERS. With all our testing, we found that this is a good ratio. This is in the ballpark, where these setting should be. Again with every benchmarking, you might find that you might have to change these values. If you are higher than you really have to be on either one of these values, you will not have any problems, because there is minimal overhead associated with each I/O server and any unused I/O servers remains idle along with any unused I/O cleaner. It is always better to have a few to many, rather than not have enough, when at peak loads.

4.14 NUM_IOSERVERS: Number of I/O servers

I/O servers are used on behalf of the database agents to perform prefetch I/O and asynchronous I/O by utilities such as backup and restore. This parameter specifies the number of I/O servers for a database. No more than this number of I/Os for prefetching and utilities can be in progress for a database at any time. An I/O server waits while an I/O operation that it initiated is in progress. Non-prefetch I/Os are scheduled directly from the database agents and as a result are not constrained by NUM_IOSERVERS.

Recommendation: Compare with the *recommendation* provided in 4.13, “NUM_IOCLEANERS: Number of async page cleaners” on page 27.

4.15 MAXFILOP: Maximum database files open per application

This database parameter specifies the maximum number of file handles that can be open for each database agent. If opening a file causes this value to be exceeded, some files in use by this agent are closed. If MAXFILOP is too small, the overhead of opening and closing files so as not to exceed this limit will become excessive and might degrade performance.

Recommendation: For LDAP, we found that the default value of 64 is too low for large LDAP directories and we increased this to 384 to start with. Adjust this, if needed, using the tools in this document.

4.16 MAXAPPLS: Maximum number of active applications

This database parameter specifies the maximum number of concurrent applications that can be connected to a database. Because each application that attaches to a database causes some private memory to be allocated, allowing a larger number of concurrent applications potentially uses more memory.

Recommendation: For LDAP large directories, we found that setting this to 100 is a good starting point that works for most all directories. We came to this after also increasing the value of the locklist to work together. Adjust this, if needed, using the tools in this document.

4.17 PKGCACHESZ: Package cache size

This database parameter is allocated out of the database shared memory, and is used for caching of sections for static and dynamic SQL statements on a database.

Recommendation: For LDAP large directories, we found that the default setting of -1 is low and that setting this to 1440 is a good starting point that works for most directories. Adjust this, if needed, using the tools in this document.

4.18 LOGFILSIZ: Size of log files

This database parameter defines the size of each primary and secondary log file. The size of these log files limits the number of log records that can be written to them before they become full and a new log file is required.

Recommendation: You must balance the size of the log files with the number of primary log files. The value of the LOGFILSIZ should be increased if the database has a large number of update, delete, or insert transactions running against it, which causes the log file to become full very quickly.

Notes:

- The upper limit of the log file size, combined with the upper limit of the number of log files (logprimary + logsecond), gives an upper limit of 256 GB of active log space.
- A log file that is too small can affect the system performance because of the overhead of archiving old log files, allocating new log files, and waiting for a usable log file.
- The value of the LOGFILSIZ should be reduced if disk space is scarce because primary logs are preallocated at this size.
- A log file that is too large can reduce your flexibility when managing archived log files and copies of log files because some media might not be able to hold an entire log file.
- If you are using log retention, the current active log file is closed and truncated when the last application disconnects from a database. When the next connection to the database occurs, the next log file is used. Therefore, if you understand the logging requirements of your concurrent applications, you might be able to determine a log file size that will not allocate excessive amounts of wasted space.

Recommendation: For large LDAP directories, we found that the default setting of 1000 is low and setting this to 5000 is a good starting point that works for most directories. Adjust this, if needed, using the tools in this document.

4.19 LOGPRIMARY: Number of primary log files

The primary log files establish a fixed amount of storage allocated to the recovery log files. This parameter allows you to specify the number of primary log files to be preallocated.

Under circular logging, the primary logs are used repeatedly in sequence, that is, when a log is full, the next primary log in the sequence is used, if it is available. A log is considered available if all units of work with log records in it have been committed or rolled back. If the next primary log in sequence is not available, then a secondary log is allocated and used. Additional secondary logs are allocated and used until the next primary log in the sequence becomes available or the limit imposed by the logsecond parameter is reached. These secondary log files are dynamically deallocated as they are no longer needed by the database manager.

Recommendation: For LDAP large directories, we found that the default setting of 3 is low and setting this to 5 is a good starting point that works for most directories. Adjust this value, if needed, using the tools in this document.

4.20 LOGSECOND: Number of secondary log files

This database parameter specifies the number of secondary log files that are created and used for recovery log files (only as needed). When the primary log files become full, the secondary log files (of size LOGFILSIZ) are allocated one at a time as needed, up to a maximum number as controlled by this parameter. An error code will be returned to the application and the database will be shut down, if more secondary log files are required than are allowed by this parameter.

Recommendation: For LDAP large directories, we found that the default setting of 2 is far too low and setting this to 60 is a good starting point that works for most directories. Adjust this, if needed, using the tools in this document.

4.21 DFT_PREFETCH_SZ: Default prefetch size

If you do not specify the prefetch size on invocation of the CREATE TABLESPACE statement, the database manager uses the current value of the DFT_PREFETCH_SZ parameter.

Recommendation: For LDAP, using the default of 32 works with most DB2 systems and should match what your extent size is set to. If you are going to be using SANs, then this value depends on your array, the extent size, or both. Using system monitoring tools, you can determine if your CPU is idle while the system is waiting for I/O.

If the prefetch size is a multiple of the extent size, the database manager might perform I/O in parallel, if the following conditions are true:

- The extents being prefetched are on different physical devices
- Multiple I/O servers are configured (NUM_IOSERVERS)

4.22 DFT_EXTENT_SZ: Default extent size of tablespaces

When a tablespace is created, EXTENTSIZE *n* can be optionally specified, where *n* is the extent size. If you do not specify the extent size on the CREATE TABLESPACE statement, the database manager uses the value given by this parameter.

Recommendation: For LDAP, using the default of 32 works with most DB2 systems and should match what your extent size is set to. If you are going to be using SANs, then this value depends on your array, the extent size, or both. Using system monitoring tools, you can determine if your CPU is idle while the system is waiting for I/O.

4.23 NEWLOGPATH: Change the database log path

This database parameter allows you to specify a string of up to 242 bytes to change the location where the log files are stored. The string can point to either a path name or to a raw device. If the string points to a path name, it must be a fully qualified path name, not a relative path name.

You should move the default location of the path of the transaction log files. By default, it is put where the DB2 instance is found, for example: /usr/opt/db2/log/LDAPDB2. It is best if you move the location to another directory, preferably on a different physical drive, if possible. This lessens the I/O contention for writing to the disk space.

If you are going to change the NEWLOGPATH, with the db2_tunings.sh, you also have to remove the comment character from the **`db2 update`** command later on in this script. *Hint*: Search for NEWLOGPATH to find the command. For example, on the NEWLOGPATH line, put in the path where the transaction logs should go.

```
NEWLOGPATH=/usr/logfiles/logs
```

Then, do a search for NEWLOGPATH again and you should find the following line. Uncomment it so that it will be used. This sets the new path to be used.

```
#db2 update db cfg for ${DBNAME} using NEWLOGPATH "${NEWLOGPATH}
```

4.24 DB2SET commands

Let us take a closer look at two DB2SET commands: DB2_PARALLEL_IO and DB2_HASH_JOIN.

4.24.1 DB2_PARALLEL_IO

This section describes how to optimize performance when data is placed on Redundant Array of Independent Disks (RAID) devices.

Procedure

You should perform the following steps for each tablespace that is stored on a RAID device:

1. Define a single container for the tablespace (using the RAID device).
2. Make the EXTENTSIZE of the tablespace equal to or a multiple of the RAID stripe size.
3. Ensure that the PREFETCHSIZE of the tablespace is:
   - The RAID stripe size multiplied by the number of RAID parallel devices (or a whole multiple of this product)

     And:
   - A multiple of the EXTENTSIZE
4. Use the DB2_PARALLEL_IO registry variable to enable parallel I/O for the tablespace.

   For LDAP, we highly recommend that you set this variable to “*” anytime you are going to be using RAID devices or putting your database instance on SANs.

   ```
   db2set DB2_PARALLEL_IO="*"
   ```

4.24.2 DB2_HASH_JOIN

The DB2_HASH_JOIN, by default, is set to *YES*. For LDAP, we highly recommend that you set this variable to *NO*. Our large directory testing produced better performance by tuning off HASH_JOINs without producing any issues with the directory or LDAP commands with this set to NO.

```
db2set DB2_HASH_JOIN=NO
```

5

Table cardinality and LDAP_MAXCARD setting

This chapter discusses how to adjust the cardinality of the table index. It also provides information about the LDAP_MAXCARD setting, which is used to influence the behavior of the DB2 search optimizer.

5.1 Adjusting table cardinality for performance

When DB2 performs a query, the DB2 optimizer defines a query plan to resolve the result set at minimal cost. In most cases, a query consists of multiple conditions. Consider the Lightweight Directory Access Protocol (LDAP) search:

```
ldapsearch -b "o=everything" objectclass=widgets
```

This translates to a DB2 query for all descendants of “o=everything” with object class of widgets. DB2 tries to determine whether it is better to find all the descendants of o=everything first, so as to determine which of these are widgets, or whether it is better to find all widgets first, so as to determine which of these are descendants of o=everything. The actual query looks something like:

```
SELECT distinct D.DEID FROM LDAPDB2.LDAP_DESC AS D WHERE D.AEID=? AND D.DEID IN
(SELECT EID FROM LDAPDB2.OBJECTCLASS WHERE OBJECTCLASS =?)
```

Note the use of the “?”. This is a parameter marker, and the actual value of the parameter marker is substituted in when the query is performed. When DB2 does the optimization, the optimizer does not know what values will be substituted for the parameter markers. Therefore, it makes an estimate based on the average values for the table. In the LDAP_DESC table, most entries have few or no descendants, because in an LDAP namespace, most entries are leaf entries. Therefore, the DB2 optimizer estimates that this clause will be very selective. However, the OBJECTCLASS table is the largest table in the directory's database, because every object has several object class attribute values. And for most object classes, there are many entries. Therefore, the DB2 optimizer estimates that the objectclass=widgets clause is not very selective.

Assume that there are few entries in the tree with an object class of “widgets” so that the result set of the original LDAP search is small. The default DB2 optimizer behavior turns out to be very expensive. It finds all the descendants of o=everything first, and then it discards all of those that do not have objectclass=widgets. The result is *bad performance*.

To remedy this, we can adjust the *cardinality* of the table index. When a DB2 **runstats** is performed, DB2 counts the number of distinct values in each indexed attribute, by comparison with the length of the table (its cardinality). If the number of distinct values is low by comparison with the cardinality, DB2 concludes that a search for this attribute value will not be very selective. If the number of distinct values is high, DB2 concludes that the search will be very selective. By setting the cardinality of the LDAP_DESC table artificially high, you can ensure that the descendant table is not queried first. The cardinality is set as follows (we do set this in the **tune_runstats.sh**, which we discuss in 7.2, “How to use tune_runstats.sh” on page 57):

```
db2 update sysstat.tables set card = 9E18 where tabname = 'LDAP_DESC'
```

After this is done, the previous query for widgets objects in the o=everything tree becomes much faster. The difference might be on the order of 100X or more.

Unfortunately, this step is not always wise. Consider the following LDAP search:

```
ldapsearch -b "ou=verySmall, o=everything" objectclass=person
```

We assume that the ou called “verySmall” contains very few entries. However, there are perhaps millions of entries in the directory with objectclass=person. Therefore, the best approach is to find the descendants of ou=verySmall first, and then find all of those with objectclass=person. This is what DB2 does by default. But, if the cardinality of the LDAP_DESC table has been set as previously described, the DB2 optimizer instead chooses to find all person entries first, and then find which of those are descendants of ou=verySmall. This makes the search much slower. Again, the difference can be in the order of 100x.

Therefore, the appropriate setting of cardinality for LDAP_DESC depends on whether most subtree searches are for small subtrees or large ones. This setting only affects subtree searches, not one level or base searches. For most LDAP usage patterns, searches of large subtrees predominate. Therefore, for most LDAP usage patterns, the cardinality of LDAP_DESC should be set as previously described.

The DB2 statistics tables are updated every time **runstats** is performed. When this happens, the artificial cardinality that was set into the table is overwritten. Therefore, whenever **runstats** is performed, the cardinality of LDAP_DESC has to be adjusted again (this is taken care of with the **tune_runstats.sh**, which is covered in 7.2, “How to use tune_runstats.sh” on page 57).

Several exception cases should be mentioned. Consider:

```
ldapsearch -b "ou=verySmall, o=everything" objectclass=*
```

This search translates to the DB2 query:

```
SELECT distinct D.DEID FROM LDAPDB2.LDAP_DESC AS D WHERE D.AEID=?
```

The OBJECTCLASS table is not queried at all, because “objectclass=*” is true for every entry in the directory. Therefore, this query performs the same way regardless of how the cardinality of the descendant table is set. Consider also:

```
ldapsearch -b "ou=verySmall, o=everything" rareAttribute=blue
```

We assume that very few entries in the directory have a value for rareAttribute. In this case, the query performs well, regardless of how the cardinality is set.

The cardinality of the LDAP_DESC table is best adjusted using the LDAP_MAXCARD setting in ibmslapd.conf. Refer to 5.2, “LDAP_MAXCARD setting” on page 36, for more information about this.

Infrequently, there are cases where it might help to set the cardinality of the LDAP_ENTRY table. This occurs whenever there are many one-level searches. Consider the query:

```
ldapsearch -b "ou=people, o=everything" -s one objectclass=widgets
```

We assume that the ou=people entry has more than a million descendants, but that very few of those are widgets. The corresponding SQL statement looks something like:

```
SELECT distinct E.EID FROM LDAPDB2.LDAP_ENTRY AS E, LDAPDB2.LDAP_ENTRY as pchild
WHERE E.EID=pchild.EID AND pchild.PEID=? AND E.EID IN (SELECT EID FROM
LDAPDB2.OBJECTCLASS WHERE OBJECTCLASS =?)
```

In this case, DB2 does not know how many immediate children the search base entry has, because of the use of the “?” placeholder. Therefore, DB2 assumes that there are few, and finds all the children first. Again, this is very expensive. To remedy this, one can set the cardinality of the entry table as very large, as follows:

```
db2 update sysstat.tables set card = 9E18 where tabname = 'LDAP_ENTRY'
```

Future releases of Tivoli Directory Server and DB2 might contain optimizations to minimize or eliminate the need to adjust cardinalities artificially. This also can be set with the **tune_runstats.sh** by un-remarking this line (line 211) in the script.

5.2 LDAP_MAXCARD setting

IBM Tivoli Directory Server has a special environment variable in ibmslapd.conf called *LDAP_MAXCARD*, which is used to influence the behavior of the DB2 search optimizer. One of the largest tables in the directory is the LDAP_DESC table. This table and its indexes are used when a Lightweight Directory Access Protocol (LDAP) subtree search is requested. The directory must ensure that all returned values are descendants of the root of the search. A subtree search typically includes a search filter. The DB2 optimizer determines whether it should first find all descendants of the root of the search and then evaluate whether they match the search filter, or whether it should evaluate the search filter first, and then evaluate whether each of the matching entries is a descendant of the root of the tree. DB2 does this using an internal statistic for the LDAP_DESC table called the *cardinality*. If LDAP_MAXCARD is on, the cardinality for LDAP_DESC is set to an artificially high number (9E18, which means 9000000000000000000). This tells the DB2 search optimizer to always use LDAP_DESC last when evaluating subtree searches.

LDAP_MAXCARD can make searches much faster or much slower, depending on usage patterns. Consider the following two searches.

- ldapsearch -b “ou=very small, ou=small, o=Acme, c=US” -s subtree objectclass=person
- ldapsearch -b “o=Acme, c=US” -s subtree objectclass=rareWidgets

If LDAP_MAXCARD is off, the DB2 optimizer will use the real cardinality of the LDAP_DESC table and the OBJECTCLASS table to figure out the best way to evaluate the search. The only table in the directory larger than the LDAP_DESC table is the OBJECTCLASS table. Therefore, if LDAP_MAXCARD is off, DB2 will always find all descendants of the root of the search first, and then it will check to see which of these are of the required object class. This approach resolves search 1 very quickly, because there are very few descendants of ou=very small. However, it works very poorly for search 2. The entry o=Acme might have millions of descendants, and each must be checked to see if it is of object class rareWidgets. Very few are.

If LDAP_MAXCARD is on, the DB2 optimizer will never use the LDAP_DESC table first, because the cardinality is too big. Instead, it finds all entries that match the filter. This works very well for search 2, because only a handful of entries are of object class rareWidgets. DB2 finds those and then checks each to find which are descendants of o=Acme. Unfortunately, this approach works very poorly for search 1. The directory might have millions of entries with object class person, and DB2 ends up evaluating each to obtain which is a descendant of ou=very small. Very few are.

The DB2 optimizer does not recognize the difference between the two statements. It does not know which search bases give a large number of descendants and which give a small number. On an average, most entries in the directory have few or no descendants, therefore the DB2 optimizer assumes, given the default cardinality, that it should evaluate the descendants first. You must determine whether searches of type 1 predominate in your directory traffic, or searches of type 2. If searches of type 1 predominate, you want LDAP_MAXCARD off. If searches of type 2 predominate, you want LDAP_MAXCARD on. This tuning has a big effect. One value of the switch can make certain searches 10000X slower than the other value, and if the setting is wrong, the server gives poor throughput with CPU consumption running near 100%.

Depending on the version of the Directory Server you are using, LDAP_MAXCARD works differently:

- In Version 5.2, it does the card tuning by default, unless the environment variable LDAP_MAXCARD is set to *NO*.
- In Version 5.2 Fix Pack 3, it does *not* do the card tuning by default, unless the environment variable LDAP_MAXCARD is set to *YES*. If you are going to set it to YES, you will also have to update the **tune_runstats.sh** (this is covered in 7.2, "How to use tune_runstats.sh" on page 57) and edit to either remark out the card line or un-remark the sysstat.tables set card line for LDAP_DESC or LDAP_ENTRY.

With 6.0, it has changed again:

- Version 6.0 by default will set LDAP_DESC card to 9E18 once only when the server starts up, if LDAP_MAXCARD is not defined.
- If LDAP_MAXCARD is set to YES, the server will set it once, every minute.
- If LDAP_MAXCARD is set to NO, the server will not set it at all.

In practice, you should always set the value to either YES or NO. There is a good reason for the server to set the cardinality every minute. If reinstates is run on the LDAP_DESC table, it sets the cardinality to the default value, therefore the Directory Server must reset it to 9E18 to re-enable LDAP_MAXCARD. For bulkloads, you should always have LDAP_MAXCARD=YES (because most directory entries have few descendants). At other times, the best setting depends on whether searches of type 1 or type 2 predominate in the load.

An additional index on the LDAP_DESC table is strongly recommended. This index is especially important if LDAP_MAXCARD is set to ON. The index is not needed, if you have LDAP_MAXCARD ON. The name of the index is LDAP_DESC_DEID. (LDAP_DESC_AEID is always created.) You can check for its presence as follows. (The commands below assume that the DB2 instance name and LDAP database name are LDAPDB2, but you should substitute correct values for your installation.)

```
su - LDAPDB2
db2 connect to ldapdb2
db2 "SELECT INDNAME FROM SYSCAT.INDEXES WHERE TABNAME='LDAP_DESC'"
```

This select statement should show two indexes, LDAP_DESC_AEID and LDAP_DESC_DEID. If the second is not present, you can create it as follows.

```
db2 "CREATE INDEX LDAPDB2.LDAP_DESC_DEID ON LDAPDB2.LDAP_DESC ('AEID' ASC, 'DEID'
ASC) MINPCTUSED 10 ALLOW REVERSE SCANS"
db2 commit
db2 connect reset
```

6

Tools and scripts

In this chapter, we introduce some important tools and scripts, some of which are available as a download from the ITSO Web site (refer to Appendix G, “Additional material” on page 193). These tools are:

- ITDSAUDIT.JAR
- tune_enablemonitor.sh
- perftune_enablemonitor_all.sh
- tune_disablemonitor.sh
- perfanalyze_indexes.pl
- perfanalyze_audit.pl
- perfanalyze_dynamicsql.pl
- perfanalyze_database.pl
- perfanalyze_tables.pl

© Copyright IBM Corp. 2007. All rights reserved.

6.1 ITDSAUDIT.JAR

Understanding what the directory is doing and how well it is doing is necessary before, during, and after any tuning exercise. Understanding the transaction types, their distribution, and the current performance of the directory is necessary to determine what (if any) tuning might be necessary. In addition, without a baseline and method to measure the impact of tuning changes, it is possible to degrade the performance of the directory or worse yet, to be unable to demonstrate or measure the effects of tuning and the actual performance of the directory post tuning.

As with any client/server application, determining where the performance issues actually reside can be troublesome and difficult to diagnose. There might be server side issues (poor performance to client requests), network throughput/stability issues, or it might be the client that is the actual culprit (for example, malformed queries or inability to process information fast enough). The Directory Server can be scoped to provide services to one or many clients and their interaction must also be taken into account when evaluating the directories performance (for example, one application consuming the majority of the CPU).

Along with the other tools discussed in this document, itdsaudit.jar provides a snapshot look at the directory and the operations executed against it by parsing and reporting the transactions as seen by the directory audit log. When audit logging is enabled, the Directory Server provides a comprehensive list of client requests, parameters, response times, and success/failure of the request for each transaction performed during the time the audit log is turned on.

6.1.1 Theory of operation

The audit log (a standard text file) provides a tremendous amount of information when enabled. Depending upon what level and options are selected, the audit log can contain:

- Directory control information (enabling auditing, audit levels, and so on)
- For each transaction (as appropriate for the transaction type):
  - Start and stop time of a transaction
  - Transaction type (bind, unbind, search, add, delete and modify)
  - Bind ID performing the transaction
  - Success or failure of the request
  - Filters
  - Controls
  - Attributes

The difficulty with the log (which can be huge, based upon the transaction volume) is extracting and turning the raw data into information we can use. The itdsaudit.jar tool can assist with this.

6.1.2 Prerequisites

itdsaudit.jar is a Java™ application packaged as an executable jar file. It has been developed against Java 1.5, but there is no known reason why it will not run with any recent version of Java. The only prerequisites are:

- Java must be available.
- The user must have read permissions for the input file.
- The location where the input audit log resides must be writeable by the invoking user if you want PDF output.

6.1.3 Invoking itdsaudit.jar

itdsaudit.jar has a very simple invocation because it has no switches and expects only the file name of the input audit log on the command line. The command line for running itdsaudit.jar is:

```
java -jar itdsaudit.jar <path_and_file_name>
```

Where:

“java -jar itdsaudit.jar” is the normal start command for the Java JVM™ and tells the JVM to start the *main* entry point of the supplied jar file name.

<path_and_file_name> is the full path and name of the audit log to be parsed. If the file name contains spaces it must be quoted to ensure that it is passed to itdsaudit.jar properly.

6.1.4 itdsaudit.jar error messages

There are three error conditions that itdsaudit.jar can run into:

- No file name passed on the command line. If this occurs, itdsaudit.jar produces an output with the following error message.

  ```
  ITDS audit log parser version 0.2b
  You must supply the file name to parse on the command line, e.g.: java -jar
  itdsaudit.jar <FILENAME>
  itdsaudit.jar will parse and report (both to stdout and a PDF file) the
  absolute and statistical information contained within an ITDS audit log file.
  Note: the PDF filename is always <FILENAME>.pdf and any existing file with that
  name will be overwritten.
  ```

- Unable to open the input file.

  ```
  Unable to open input file: bad_name.log. halting execution. Make sure you quote
  the filename if it has spaces in it.
  ```

- Unable to write the PDF file. Because itdsaudit.jar writes the PDF file to the same directory where the input file is located, if it is unable to create the file there, it will print an error message and continue with stdout output only.

  ```
  Unable to open PDF file for output (make sure you have write privileges where
  the input file exists). Continuing with stdout only.
  ```

6.1.5 itdsaudit.jar stdout output

itdsaudit.jar outputs to stdout general statistical information found within the audit log. The format and output looks similar to the following (depending upon the information contained within the audit log) as shown in Example 6-1.

Example 6-1 itdsaudit.jar stdout output

```
-------------------------- Totals -------------------------
Total TransactionTime (hh:mm: ss.sss)     = 2:7:23.678 Note: Not clock time
Total Number of Transactions            = 42332
Average Transaction time in Milliseconds = 180
-------------------------- Binds --------------------------
Total Number of Binds                   = 6
Average Bind time in Milliseconds       = 1
```

```
-------------------------- Unbinds ------------------------
Total Number of Unbinds                 = 6
Average Unbind time in Milliseconds     = 0
-------------------------- Searches ------------------------
Total Number of Searches                = 41406
Average Search time in Milliseconds     = 183
Longest Search (time in Milliseconds)   = 18947
Longest Search text
AuditV3--2006-03-09-15:14:46.990-05:00--V3 Search--bindDN: cn=Directory Manager--client:
166.86.124.79:62309--connectionID: 98--received: 2006-03-09-15:14:28.043-05:00--Success
base: ou=orgChart, erglobalid=00000000000000000000, ou=abccompany, dc=itim
scope: singleLevel
derefAliases: derefAlways
typesOnly: false
filter:
(&(erparent=ERGLOBALID=6776123625579741330,OU=ORGCHART,ERGLOBALID=00000000000000000000,OU=abccom
pany,DC=ITIM)(|(objectclass=ERBPORGITEM)(objectclass=ERLOCATIONITEM)(objectclass=ERORGUNITITEM)(
objectclass=ERSECURITYDOMAINITEM)))
attributes: erparent, objectclass,
Total # of Attributes used in search filters = 16
Attribute = erpolicytarget Attribute count = 9772
Attribute = erprerequisite Attribute count = 21
Attribute = erpolicymembership Attribute count = 11386
Attribute = ou Attribute count = 4
Attribute = objectclass Attribute count = 47838
Attribute = owner Attribute count = 1264
Attribute = businesscategory Attribute count = 3660
Attribute = uid Attribute count = 613
Attribute = erobjectprofilename Attribute count = 3662
Attribute = erreqpolicytarget Attribute count = 10077
Attribute = erword Attribute count = 921
Attribute = erenabled Attribute count = 10686
Attribute = cemploymentstatus Attribute count = 610
Attribute = erisdeleted Attribute count = 24
Attribute = erparent Attribute count = 14958
Attribute = erservicename Attribute count = 1831
-------------------------- Adds -----------------------------
Total Number of Adds                    = 0
Average Add time in Milliseconds        = 0
-------------------------- Deletes --------------------------
Total Number of Deletes                 = 0
Average Delete time in Milliseconds     = 0
-------------------------- Modifies --------------------------
Total Number of Modifies                = 914
Average Modify time in Milliseconds     = 35
-------------------------- Unknowns --------------------------
Total Number of Unknown                 =
Average Unknown time in Milliseconds    =
```

6.1.6 itdsaudit.jar PDF output

itdsaudit.jar's PDF output contains additional information and output in a format that is appropriate for inclusion in tuning before, during, and after documentation. The PDF document is broken down into a number of sections.

Title page

The title page contains:

- The version of itdsaudit.jar
- The input file name
- The date and timestamp for the creation of the PDF (useful for tracking and reporting purposes)

Summary section

The summary section contains multiple charts and tables on the information found:

- Transaction types and totals

  Table 6-1 provides a summary count of the transactions and their average time to complete (in millisecond (ms)). As can be seen here, the overall average transaction time was 183 ms (very poor), but the average for binds, unbinds, and modifies are within norms for the hardware platform the directory was running on.

  Table 6-1 Transaction totals and types

Transaction types and totals		
Operation	**Count**	**Average (ms)**
Binds	6	1
Unbinds	6	0
Searches	41406	183
Adds	0	0
Deletes	0	0
Modifies	914	35
Totals	42332	180

  This is shown in a graph format in Figure 6-1 (this is the same information, but it shows at a glance the distribution by transaction types).

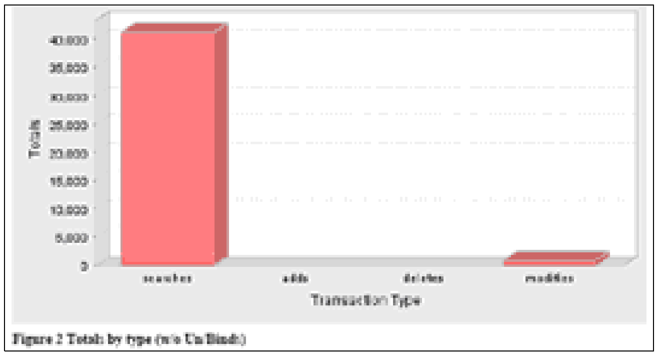


Figure 6-1 Totals by type

- Transaction types and totals without binds and unbinds

  Because binds and unbinds can skew the average transaction rate (they are normally low overhead operations and if the directory is serving as an authentication engine, they can hide issues with other transaction types). Refer to Table 6-2.

Table 6-2 Totals by type (without unbinds and binds)

Transaction types and totals		
Operation	**Count**	**Average (ms)**
Searches	41406	183
Adds	0	0
Deletes	0	0
Modifies	914	35
Totals	42332	180

This is shown in a graph format in Figure 6-2 (this is the same information, but it shows at a glance the distribution by transaction types).

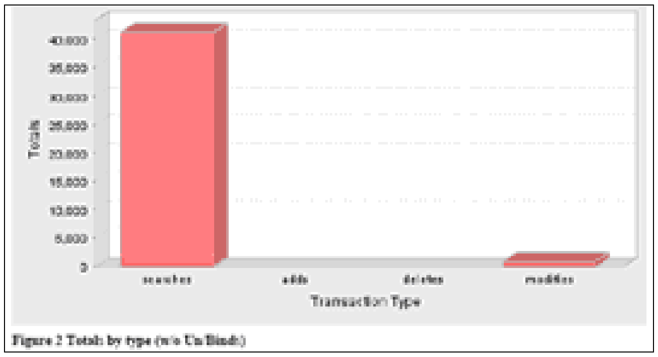


Figure 6-2 Totals by type (without unbinds and binds)

- Distribution by time (all transaction types)

  This chart in Figure 6-3 shows by transaction count the number of transactions that are completed within a given time frame. Looking at the chart, you can see that while the overall transaction time average is 183 ms, the majority of transactions completed in 10 ms or less. It was 377 transactions pulling back the entire object tree that drove the average to 183 ms.

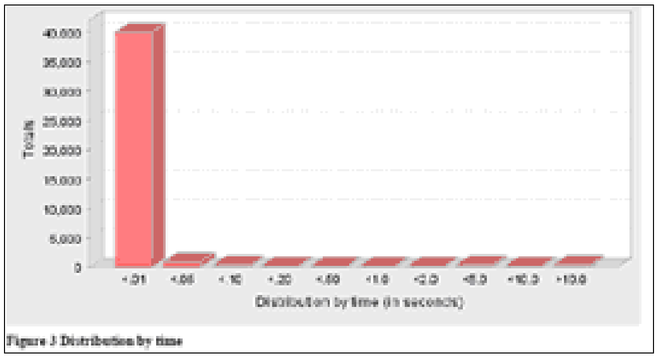


Figure 6-3 Distribution by time

Bind summary

The bind summary section provides a breakdown of the quantity and distribution by time of the bind transactions found within the audit log.

Note: Binds should be very fast low overhead transactions for the directory to process. Binds that take excessive time (> 5 ms) point to issues with directory load, performance, or possibly the network communication (for example, very slow or high error rate link).

- Bind distribution by time:

  The chart in Figure 6-4 shows by transaction count the number of transactions that completed within a given time frame.

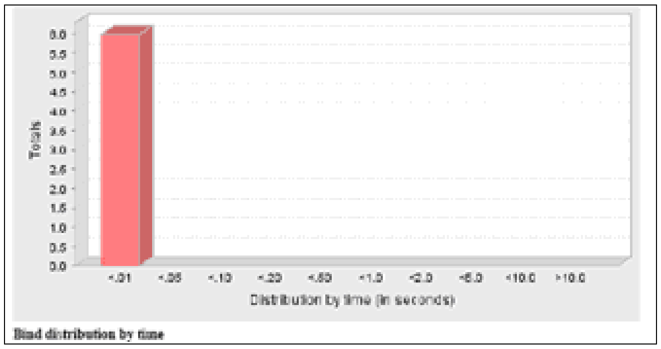


Figure 6-4 Number of transactions that completed within a given time frame

Unbind summary

The unbind summary section provides a breakdown of the quantity and distribution by time of the unbind transactions found within the audit log.

Note: Unbinds should be very fast low overhead transactions for the directory to process. Binds that take excessive time (> 5 ms) point to issues with directory load or performance.

Add summary

The add summary section provides a breakdown of the quantity and distribution by time of the add transactions found within the audit log.

The chart in Figure 6-5 shows by transaction count the number of transactions that completed within a given time frame.

Note: Because this audit log had no adds, the chart output defaults to centering a zero input across the chart. This is not an error.

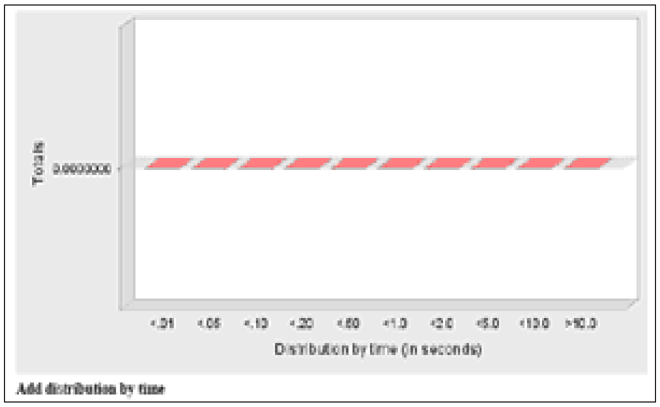


Figure 6-5 Add distribution by time

Search summary

The search summary section provides a breakdown of the quantity and distribution by time of the search transactions found within the audit log.

The chart in Figure 6-6 shows by transaction count the number of transactions that completed within a given time frame.

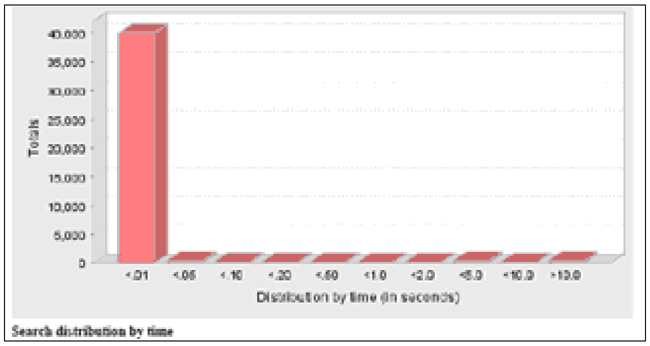
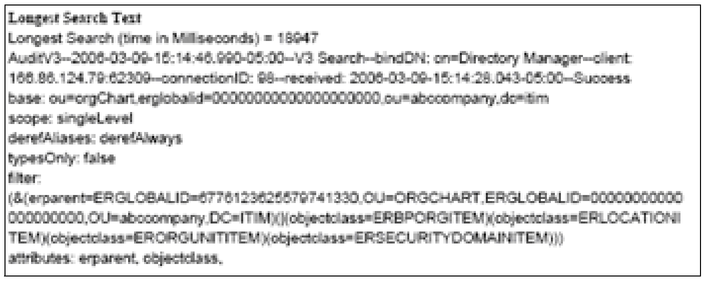


Figure 6-6 Search distribution by time

In addition to the distribution chart, the search section (if searches were included in the audit log) also shows two additional pieces of information in the output:

- The longest search text

  The search query that took the longest time to complete is reproduced in the document to allow you to evaluate and determine if tuning or client action has to be taken. See Figure 6-7.

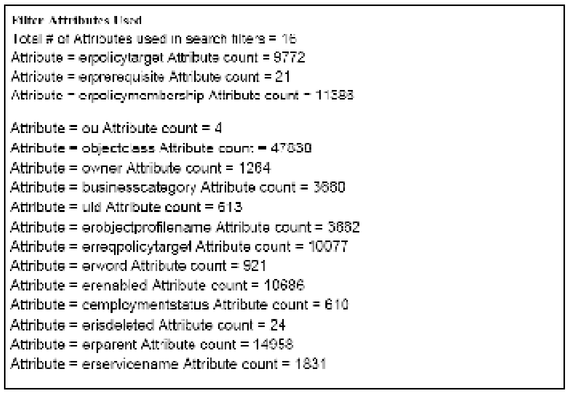


```
Longest Search Text
Longest Search (time in Milliseconds) = 18947
AuditV3--2006-03-09-15:14:46.990-05:00--V3 Search--bindDN: cn=Directory Manager--client:
166.86.124.79:62309--connectionID: 98--received: 2006-03-09-15:14:28.043-05:00--Success
base: ou=orgChart,erglobalid=00000000000000000000,ou=abccompany,dc=itim
scope: singleLevel
derefAliases: derefAlways
typesOnly: false
filter:
(&(erparent=ERGLOBALID=6776123625579741330,OU=ORGCHART,ERGLOBALID=0000000000
0000000000,OU=abccompany,DC=ITIM)(|(objectclass=ERBPORGITEM)(objectclass=ERLOCATIONI
TEM)(objectclass=ERORGUNITITEM)(objectclass=ERSECURITYDOMAINITEM)))
attributes: erparent, objectclass,
```

Figure 6-7 Longest search text

- Filter attributes used

  A summary of all the attributes used in every search transaction found within the audit log. This can be useful to determine if indexes have to be built or enabled. See Figure 6-8.



```
Filter Attributes Used
Total # of Attributes used in search filters = 16
Attribute = erpolicytarget Attribute count = 9772
Attribute = erprerequisite Attribute count = 21
Attribute = erpolicymembership Attribute count = 11388

Attribute = ou Attribute count = 4
Attribute = objectclass Attribute count = 47830
Attribute = owner Attribute count = 1264
Attribute = businesscategory Attribute count = 3680
Attribute = uid Attribute count = 613
Attribute = erobjectprofilename Attribute count = 3682
Attribute = erreqpolicytarget Attribute count = 10077
Attribute = erword Attribute count = 921
Attribute = erenabled Attribute count = 10686
Attribute = cemploymentstatus Attribute count = 610
Attribute = erisdeleted Attribute count = 24
Attribute = erparent Attribute count = 14958
Attribute = erservicename Attribute count = 1831
```

Figure 6-8 Filter attributes used

Delete summary

The delete summary section provides a breakdown of the quantity and distribution by time of the delete transactions found within the audit log.

The chart in Figure 6-9 shows by transaction count the number of transactions that completed within a given time frame.

Note: Because this audit log had no deletes, the chart output defaults to centering a zero input across the chart. This is not an error.

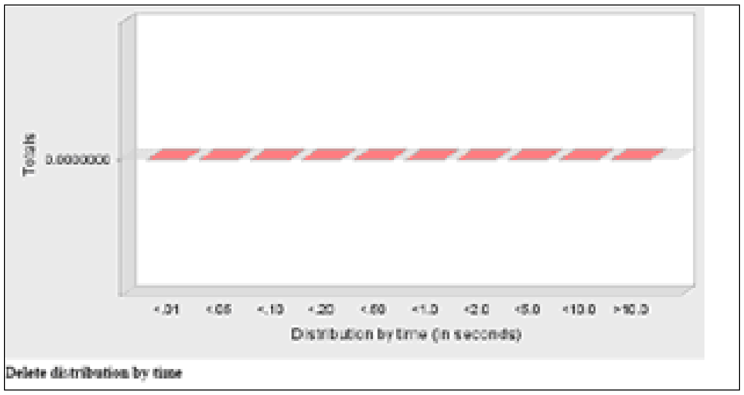


Figure 6-9 Delete distribution by time

Modify summary

The modify summary section provides a breakdown of the quantity and distribution by time of the modify transactions found within the audit log.

The chart in Figure 6-10 shows by transaction count the number of transactions that completed within a given time frame.

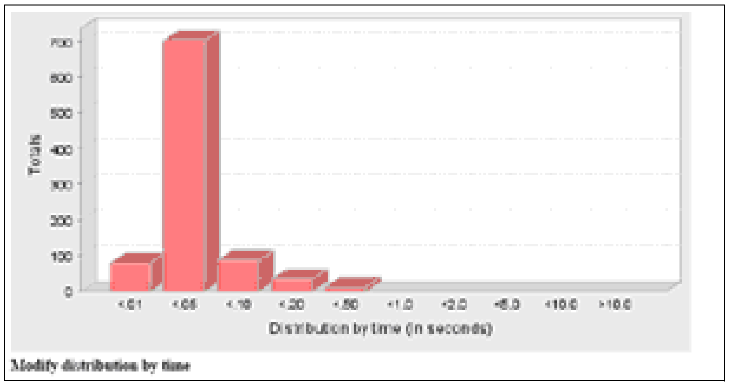


Figure 6-10 Modify distribution by time

6.2 tune_enablemonitor.sh

This script turns on the DB2 monitors in Table 6-3, which you will need running to gather statistics in the database where you can use the following scripts to read and analyze this data.

Table 6-3 DB2 monitors

Monitor switch	DBM parameter	Information provided
BUFFERPOOL	DFT_MON_BUFPOOL	Number of reads and writes, time taken
LOCK	DFT_MON_LOCK	Lock wait times, deadlocks
SORT	DFT_MON_SORT	Number of heaps used, sort performance
STATEMENT	DFT_MON_STMT	Start/stop time, statement identification
UOW (unit of work)	DFT_MON_UOW	Start/end times, completion status
TIMESTAMP	DFT_MON_TIMESTAMP	Timestamps

These monitors capture most of what you need and can be left on for long periods of time to gather data for benchmarking.

With this script, you have to stop and restart DB2 after you run the script. Make sure that Lightweight Directory Access Protocol (LDAP) is down before running this script. You can bring LDAP back up after you run this script. You must be the DB2 instance owner to run this script.

6.3 perftune_enablemonitor_all.sh

This script turns on all of the monitors listed in Table 6-3 along with the one in Table 6-4.

Table 6-4 DB2 monitor

Monitor switch	DBM parameter	Information provided
TABLE	DFT_MON_TABLE	Measure of activity (rows read/written)

This monitor impacts the performance if it is left running for long periods of time. It should be turned on only for troubleshooting and checking the tablespace.

With this script, you have to stop and restart DB2 after you run the script. Make sure LDAP is down first before running this script. You can bring LDAP back up after you run this script. You must be the DB2 instance owner to run this script. Remember that this script includes the table monitor that will impact the performance if it is left on for long periods of time.

6.4 tune_disablemonitor.sh

This script turns off all the monitors (except timestamps that have to be on all the time to keep track of events) started with either of the two monitor scripts described in the previous sections.

With this script, you have to stop and restart DB2 after you run the script. Make sure LDAP is down before running this script. You can bring LDAP back up after you run this script. You must be the DB2 instance owner to run this script.

6.5 perfanalyze_indexes.pl

This script is designed to determine if searches against the LDAP server are using filters against attributes that do not have indexes. It does this by doing the following actions:

- Locating the LDAP schema files by reading them from the ibmslapd.conf file, looking for the default schema files from a specified directory, reading the schema files to process from the command line, or reading the schema from the LDAP server directly.
- Processing the LDAP schema files to read all attributes and determining those that have indexes.
- Processing either an audit log or a dynamic SQL snapshot from the database instance to determine all attributes being searched against.
- Comparing the searched attributes against the schema attributes.
- Printing a report of the attributes search on and their index status.

Because the script requires the LDAP schema files, it is easiest to run it from the Directory Server itself. If this is not possible, or not desirable, the user can specify which directory on the local machine contains the schema files or the schema can be read from a remote server. If attributes are encountered in the input but were not found in the LDAP schema (usually due to a missing schema file that the Directory Server is using) the report will mention this fact, but will be unable to determine the index status of these attributes.

Not every searched attribute needs an index. Indexes create additional work for DB2 when inserting or modifying entries with indexed attributes. Attributes that are searched on very rarely might not benefit from being indexed given the overhead required to maintain them. The script attempts to include the number of times the attribute was searched on if available to aid in the evaluation if an index is necessary.

6.5.1 Usage

Invoking the script with the “-h” option produces an output with the following usage information:

```
perfanalyze_indexes.pl [-c confFile | -d schemaDir | -s schemaFiles -r
[remoteHost] [-i inputFile] [-o outputFile] [-a]
```

- Schema options:

-c	Fully qualified path name to ibmslapd.conf file
-d	Directory where schema files are located
-s	Semicolon separated list of schema files (no spaces)
-r	Remote host to pull schema from

- Other options:

-i	File containing dynamic SQL statements or audit log for processing
-o	File to put the processed results in, default is STDOUT
-a	Print all attributes searched on, not just un-indexed ones

Note: Use UNIX-style slashes for all files and directories, even if on Windows. If no arguments are given, the program will read input from STDIN.

6.5.2 Examples

Use the following script to analyze indexes from an audit log in the current directory based on the schema pulled from the ibmslapd.conf file on a Windows platform. The -a option shows all attributes, not just un-indexed ones:

```
perfanlayze_indexes.pl -c "c:/program files/ibm/ldap/etc/ibmslapd.conf" -i
audit.log -a
```

To analyze indexes from a dynamic SQL snapshot with the default schema files in a given directory but include a custom file as well:

```
perfanlayze_indexes.pl -d /usr/ldap/etc -s /usr/ldap/etc/custom.schema -i
idsdb2.snapshot
```

To analyze indexes from an audit log based on a schema from a remote machine:

```
perfanlayze_indexes.pl -r ldapserver -i audit.log
```

6.6 perfanalyze_audit.pl

This script is designed to find metadata from information in an audit log such as what queries occur most frequently and how long queries take. It can do this for either the full query with the searched on values or in a *fuzzy* method where the attributes are retained but the value searched on is discarded. The fuzzy method is useful for determining what kind of filter queries are being used regardless of the values for attributes in the filter.

Use this script to determine which LDAP queries are taking the longest, possibly from missing indexes. Use this in conjunction with the perfanalyze_indexes.pl script to locate and index attributes used in long-running LDAP queries.

6.6.1 Usage

Invoking the script with the “-h” option produces an output with the following usage information:

```
perfscripts-devel/perfanalyze_audit.pl [ -i inputFile ] [ -o outputFile ] [ -f
filterMethod ] [ -t [ -c cutOff ] | -g | -s | -d | -b | -p | -m timeFrame ]
```

- Filter options:

-f	Filter method; the following options are valid:	
all		All filters, do not collect similar filters
fuzzy		Use fuzzy filters (for example, no attribute values), default
full		Use full filters

- Output options:

-t	Show search filter timings
-d	Show search distribution timings
-s	Show transaction summary
-g	Show search filter frequencies
-m	Show time-interval stats, timeFrame is one of: second, minute, hour, day, month
-c	Statements longer than this time are not included in timings report, default is 0.1
-b	Show search bases
-p	Show search scopes

- Other options:

-h	Displays this help information
-i	File containing log for processing
-o	File to put the processed results in, default is STDOUT

If no inputFile is given, the program will read input from STDIN.

6.6.2 Examples

To see what queries are taking a long time to run regardless of the values in the filter (for example, fuzzy):

```
perfanlayze_audit.pl -i audit.log -f fuzzy
```

To see what queries are taking a long time to run including the values in the filter (for example, full):

```
perfanlayze_audit.pl -i audit.log -f full
```

To see the frequency of queries in the log:

```
perfanlayze_audit.pl -i audit.log -g
```

To see all queries and a transaction summary:

```
perfanlayze_audit.pl -i audit.log -s -f all
```

6.7 perfanalyze_dynamicsql.pl

This script is designed to locate slow queries from a dynamic SQL database snapshot. The dynamic SQL snapshot shows how long a query takes to execute in aggregate form. This is less meaningful from a response-time perspective than how long each individual query took to execute. This script provides a list of queries sorted by the execution time and the number of times those queries were executed. Long-running queries that are seen very infrequently are generally less of a problem than longish queries that are seen frequently.

Use this script to determine which LDAP queries are taking the longest, possibly from missing indexes. Use this in conjunction with the perfanalyze_indexes.pl script to locate and index attributes used in long-running LDAP queries.

Because SQL queries can be long, the printed report is truncated for the screen. To view the entire query (or to change the length of the SQL reported), use the -t option. A zero -t value results in the query being printed in full.

By default, only the queries taking longer than 0.1 seconds are shown. To change the cut-off value, use the -c option. A zero -c value results in all queries being printed.

If run on a UNIX machine as the database owner, the script can pull the dynamic SQL snapshot itself using the -d flag. If you want to retain this snapshot, use the -s flag as well.

6.7.1 Usage

Invoking the script with the “-h” option produces an output with the following usage information:

```
perfanalyze_dynamicsql.pl [-i inputFile | [-d databaseName | -s]] [-o outputFile]
```

-i	File containing dynamic SQL statements for processing
-d	Database name to get dynamic SQL statements from directly
-s	If given the temporary file with the dynamic SQL will be saved
-o	File to put the processed results in, default is STDOUT
-t	Length to truncate statement at, default is 90 characters; 0 = do not truncate
-c	Time cut-off; statements longer than this time are not included, default is 0.1
-r	Column to sort by, default is secPerExec
-n	Include queries that have no statistics information
-w	Include the number of rows written

If no arguments are given, the program will read input from STDIN.

6.7.2 Examples

To get a list of queries sorted by execution time:

```
perfanlayze_dynamicsql.pl -i idsdb2.snapshot
```

To view all queries regardless of execution time:

```
perfanlayze_dynamicsql.pl -i idsdb2.snapshot -c0
```

To view the full SQL for queries taking longer than 0.01 seconds:

```
perfanlayze_dynamicsql.pl -i idsdb2.snapshot -c0.01 -t0
```

6.8 perfanalyze_database.pl

This script is designed to parse and report on the database by analyzing database and/or buffer pool snapshots and producing statistical information to stdout and a file. Maximizing the hit ratio on the buffer pools provides a tremendous increase in performance of the database (to the extent possible). Using this script enables you to determine what hit ratio is currently being seen and what impact increasing buffer pools has on it. The ultimate goal is to obtain a ratio of 99% on all buffer pool hits.

If run on a UNIX machine as the database owner, the script can pull the snapshots directly using the -d flag. If you want to retain this snapshot, use the -s flag as well.

6.8.1 Usage

Invoking the script with the "-h" option produces an output with the following usage information:

```
Usage: $0 [-i inputFile | [-d databaseName | -s]] [-o outputFile]
```

-i	File containing snapshot for processing
-d	Database name to get snapshot from directly
-s	If given the temporary file with the snapshot will be saved
-o	File to put the processed results in, default is STDOUT

If no arguments are given, the program will read input from STDIN.

6.8.2 Examples

To get a report from a database snapshot file:

```
perfanlayze_database.pl -i idsdb2.snapshot
```

To get a report directly from the database:

```
perfanlayze_database.pl -d DATABASENAME
```

To save the report (either from a file or from the database directly):

```
perfanlayze_database.pl -i idsdb2.snapshot -s -o SAVEFILENAME
```

6.9 perfanalyze_tables.pl

This script is designed to parse and report on DB2 table information. The information returned includes rows read, written, overflows, and page reorgs.

If run on a UNIX machine as the database owner, the script can pull the snapshots directly using the -d flag. If you want to retain this snapshot, use the -s flag as well.

6.9.1 Usage

Invoking the script with the "-h" option produces an output with the following usage information:

```
Usage: $0 [-i inputFile | [-d databaseName | -s]] [-o outputFile]
```

-i	File containing snapshot for processing
-d	Database name to get snapshot from database directly
-s	If given the temporary file with the snapshot will be saved
-o	File to put the processed results in, default is STDOUT
-r	Column to sort by, default is rowsRead

If no arguments are given, the program will read input from STDIN.

6.9.2 Examples

To get a report from a database snapshot file:

```
perfanlayze_tables.pl -i idsdb2table.snapshot
```

To get a report directly from the database:

```
perfanlayze_tables.pl -d DATABASENAME
```

To save the report (either from a file or from the database directly):

```
perfanlayze_tables.pl -i idsdb2table.snapshot -s -o SAVEFILENAME
```

7

RUNSTATS: Why you have to run this

DB2 uses a sophisticated set of algorithms to optimize the access to data stored in a database. These algorithms depend upon many factors, including the organization of the data in the database, and the distribution of that data in each table. Distribution of data is represented by a set of statistics maintained by the database manager.

In addition, IBM Tivoli Directory Server creates a number of indexes for tables in the database. These indexes are used to minimize the data accessed in order to locate a particular row in a table.

In a read-only environment, the distribution of the data changes very little. However, with the environment that has a large number of updates and additions to the database on a daily basis, it is common for the distribution of the data to change significantly. Similarly, it is quite possible for data in tables to become ordered in an inefficient manner.

To remedy these situations, we have provided for you a script that can help optimize the access to data by updating the statistics and to reorganize the data within the tables of the database. The script is called *tune_runstats.sh*, and it is covered in 7.2, "How to use tune_runstats.sh" on page 57. This script can be set up with a cron job to run every night if you have a high rate of changes to your database, or weekly if you have a low rate of changes. This script can be executed with Lightweight Directory Access Protocol (LDAP) up and running. It will be good to run the script at least once a month with LDAP shut down and restart. This should clean out any index issues.

Along with running `tune_runstats.sh`, there will be times when just running this script will not give you all the performance that you should be getting. This is the time when you have to run a procedure called *DB2 Reorg*, which we cover in more detail in Chapter 8, "REORG: When and how you should run this" on page 59.

© Copyright IBM Corp. 2007. All rights reserved.

7.1 Optimization

The optimizer uses the catalog tables from a database to obtain information about the database, the amount of data in it, and other characteristics, and uses this information to choose the best way to access the data. If current statistics are not available, the optimizer might choose an inefficient access plan based on inaccurate default statistics.

Optimizing the database updates the statistics related to the data tables, which improves performance and query speed. Optimizing the database periodically or after heavy database updates by tuning the organization of the data in DB2 by using the **`tune_runstats.sh`** and **`reorg`** commands are very important for optimal performance.

The **`tune_runstats.sh`** command updates statistical information to the DB2 optimizer to improve performance, and reports statistics on the organization of the database tables.

We use the following options for **`runstats`** on a Version 8x of DB2:

- WITH DISTRIBUTION ON ALL COLUMNS AND SAMPLED DETAILED INDEXES ALL

  This statement breaks down to the following:

  – Distribution Clause: WITH DISTRIBUTION
  – On Dist Cols Clause: ON ALL COLUMNS
  – Index Clause: SAMPLED DETAILED INDEXES ALL

- AND INDEXES

  Collects and updates statistics for both the table and the indexes.

- SAMPLED

  This option, when used with the DETAILED option, allows RUNSTATS to employ a CPU sampling technique when compiling the extended index statistics. If the option is not specified, every entry in the index is examined to compute the extended index statistics. If the directory is large, most tables in the directory are also large, therefore the SAMPLED option must be used or **`runstats`** will take too long. However, some tables might be small. For small tables, it might be best not to use SAMPLED, as it gives inaccurate information.

- ON ALL COLUMNS

  Statistics collection can be done on some columns and not on others. Columns such as LONG VARCHAR or CLOB columns are ineligible. If it is required to collect statistics on all eligible columns, you can use the ON ALL COLUMNS clause. Columns can be specified either for basic statistics collection (on-cols-clause) or in conjunction with the WITH DISTRIBUTION clause (on-dist-cols-clause). The ON ALL COLUMNS specification is the default option if neither of the column-specific clauses is specified.

  If it is specified in the on-cols-clause, all columns will have only basic column statistics collected unless specific columns are chosen as part of the WITH DISTRIBUTION clause. Those columns specified as part of the WITH DISTRIBUTION clause will also have basic and distribution statistics collected.

  If the WITH DISTRIBUTION ON ALL COLUMNS is specified, both basic statistics and distribution statistics are collected for all eligible columns. Anything specified in the on-cols-clause is redundant and therefore not necessary.

We have included a *reorgchk* within the **`tune_runstats.sh`** to print out an output of the status of each of the tables and indexes.

We use the following options for reorgchk:

- CURRENT STATISTICS ON TABLE ALL

  This statement breaks down to the following:

  – CURRENT STATISTICS: Uses the current table statistics to determine if table reorganization is required.

  – ON TABLE

    - USER: Checks the tables that are owned by the runtime authorization ID.
    - SYSTEM: Checks the system tables.
    - ALL: Checks all user and system tables.

7.2 How to use tune_runstats.sh

To use **`tune_runstats.sh`**:

1. Update the DB2 statistics to improve runtime performance on *all* LDAP servers.

   ```
   su - ldapdb2
   $ ./tune_runstats.sh
   exit
   ```

   This keeps the updates current with DB2 and improves the database performance. The LDAP replication process does not include the propagation of database optimizations, therefore it requires you to run this on each of the LDAP servers you have.

2. Recycle the LDAP process each week on *all* LDAP servers, to update all indexes.

   a. Find the pid process for slapd and kill the process.

   b. Make sure the slapd process is not running.

   ```
   su - ldapdb2
   $ ./tune_runstats.sh
   $ exit
   ```

   c. Start the slapd process back up:

   ```
   $ slapd
   ```

8

REORG: When and how you should run this

The **reorg** command, using the data generated by **tune_runstats.sh**, reorganizes tablespaces to improve access performance and reorganizes indexes so that they are more efficiently clustered. The **tune_runstats.sh** (covered in 7.2, “How to use tune_runstats.sh” on page 57) and **reorg** commands can improve both search and update operation performance.

8.1 Performing a reorg as required

After you have generated organizational information about the database using **tune_runstats.sh** for reorganization, **reorg** finds the necessary tables and indexes and attempts to reorganize them. This can take a long time. The time it takes to perform the reorganization process increases as the DB2 database size increases. This step is done if **tune_runstats.sh** does not get the required results.

In general, reorganizing a table takes more time than updating statistics. You should update statistics first, and only perform reorgs on specific tables if the performance is still not as expected. Therefore, performance might be improved significantly by updating statistics first.

If you notice that your performance is not improving after running **tune_runstats.sh** and you can trend this, then this is a good time to plan for doing some reorgs of tables and maybe some indexes as needed in a maintenance window. Then rerun the **tune_runstats.sh** after you finish your reorgs (this is a requirement to update the statistics and set the cardinality back that gets reset when you do a reorg).

- Check if you have the last /tmp/tune_runstats.log from the last time you ran **tune_runstats.sh** on your Lightweight Directory Access Protocol (LDAP) server. If you do not have this, you will have to rerun the **tune_runstats.sh** to have this output log.
- The tune_runstats.log report has two sections. The first section is the table information and the second section is the indexes. An asterisk in the last column indicates a possible need for reorganization.

8.1.1 Reorg a table

To reorganize the tables with an asterisk in the last column, issue the DB2 command, as shown in the following steps:

1. Find the pid process for slapd and kill the process.
2. Make sure the slapd process is not running.
3. Execute the following commands to reorg a table.

   ```
   su - ldapdb2
   db2 connect to ldapdb2
   db2 reorg table <table_name>
   ```

4. After all reorgs are done, run the following script (required):

   ```
   su - ldapdb2
   $ ./tune_runstats.sh
   $ exit
   ```

5. Start the slapd process back up.

   ```
   $ ibmslapd
   ```

   <table_name> specifies the name of the table to be reorganized. Take a look at our example below.

   ```
   su - ldapdb2
   db2 connect to ldapdb2
   db2 reorg table LDAPDB2.IFMIT
   db2 reorg table LDAPDB2.IFMIURL
   db2 reorg table LDAPDB2.INSTALLDATE
   ```

8.1.2 Reorg an index

To reorganize database indexes with an asterisk in the last column, issue the following DB2 command:

1. Find the pid process for slapd and kill the process.
2. Make sure the slapd process is not running.
3. Run the following commands to reorg a table.

   ```
   su - ldapdb2
   db2 connect to ldapdb2
   db2 reorg table <table_name> index <index_name>
   ```

4. After all reorgs are done, run the following script (required):

   ```
   su - ldapdb2
   $ ./tune_runstats.sh
   $ exit
   ```

5. Start the slapd process back up.

   ```
   $ ibmslapd
   ```

   <table_name> specifies the name of the table and the <index_name> is the name of the index of that table that will be reorganized. To get the <index_name>: Take the two names of the index that are right next to each other and take out the space between them and put in a "." so that it looks like the following:

   Before:

   ```
   Table: LDAPDB2.TELEPHONENUMBER
   LDAPDB2 RTELEPHONENUMBER
   ```

   After:

   ```
   LDAPDB2.TELEPHONENUMBER = <tabel_name>
   LDAPDB2.RTELEPHONENUMBER = <index_name>
   ```

   Based on the example, we use the following commands:

   ```
   su - ldapdb2
   db2 connect to ldapdb2
   db2 reorg table LDAPDB2.TELEPHONENUMBER index LDAPDB2.RTELEPHONENUMBER
   db2 reorg table LDAPDB2.TELEPHONENUMBER index LDAPDB2.TELEPHONENUMBER
   db2 reorg table LDAPDB2.TELEPHONENUMBER index LDAPDB2.TELEPHONENUMBERi
   ```

   The following three indexes (or any other index not listed here, that might be created in these three tables) should never have to be reorganized, because you will require a 32 k temp page area to do it in and it will not buy you anything by doing so. These are the biggest indexes in the directory that are always changing:

   ```
   Table: LDAPDB2.LDAP_DESC
   LDAPDB2   LDAP_DESC
   Table: LDAPDB2.LDAP_ENTRY
   LDAPDB2   LDAP_ENTRY
   Table: LDAPDB2.OBJECTCLASS
   LDAPDB2   OBJECTCLASS
   ```

6. Remember that after you do all your reorgs of both tables and/or indexes, you *must* run the **`tune_runstats.sh`** again before you restart your LDAP. Example 8-1 shows a sample /tmp/tune_runstats.log.

Example 8-1   Sample tune_runstats.log

```
Table statistics:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (Effective Space Utilization of Data Pages) > 70
F3: 100 * (Required Pages / Total Pages) > 80

SCHEMA    NAME                  CARD    OV    NP    FP ACTBLK    TSIZE F1 F2 F3 REORG
----------------------------------------------------------------------------------------
...
Table: LDAPDB2.IFMIT
LDAPDB2   IFMIT                   27     0     1     2      -     1269   0 31 50 -**
Table: LDAPDB2.IFMIURL
LDAPDB2   IFMIURL                 17     0     2     3      -     4369   0 54 66 -**
Table: LDAPDB2.INSTALLDATE
LDAPDB2   INSTALLDATE              1     0     1     2      -       24   0   0 50 -**
Table: LDAPDB2.ITDSRDBMHISTORY
LDAPDB2   ITDSRDBMHISTORY          4     0     1     1      -      176   0   - 100 ---
Table: LDAPDB2.LAUNCHABLE
LDAPDB2   LAUNCHABLE           26226     0   118   128      -   472068   0 92 92 ---
...
----------------------------------------------------------------------------------------

Index statistics:

F4: CLUSTERRATIO or normalized CLUSTERFACTOR > 80
F5: 100 * (KEYS * (ISIZE + 9) + (CARD - KEYS) * 5) / ((NLEAF - NUM EMPTY LEAFS -1) *
(INDEXPAGESIZE - 96) > MIN (50, (100- PCTFREE))
F6: (100 - PCTFREE) * (FLOOR [(100 - min (10, pctfree)) / 100 * (indexPageSize - 96) / (ISIZE +
12)] ** (NLEVELS - 2)) * (indexPageSize - 96) / (KEYS * (ISIZE + 9) + (CARD - KEYS) * 5) < 100
F7: 100 * (NUMRIDS DELETED / (NUMRIDS DELETED + CARD)) < 20
F8: 100 * (NUM EMPTY LEAFS / NLEAF) < 20

SCHEMA   NAME              CARD  LEAF ELEAF LVLS ISIZE NDEL   KEYS F4 F5 F6 F7 F8 REORG
------------------------------------------------------------------------------------------------
...
Table: LDAPDB2.TARGETSERVICE
LDAPDB2 RTARGETSERVICE    26226   233   0   3   21   3 26226 95 84 49   0   0 -----
LDAPDB2 TARGETSERVICE     26226   233   0   3   21   3 26226 95 84 49  0  0 -----
LDAPDB2 TARGETSERVICEI    26226    95   0   2    4   3 26226 100 90  1  0  0 -----
Table: LDAPDB2.TELEPHONENUMBER
LDAPDB2  RTELEPHONENUMBER    1   1   0   1   15   1    1 100  -  -  50  0 ---*-
LDAPDB2  TELEPHONENUMBER     1   1   0   1   15   1    1 100  -  -  50  0 ---*-
LDAPDB2  TELEPHONENUMBERI    1   1   0   1    4   8    1 100  -  -  88  0 ---*-
...
```

CLUSTERRATIO or normalized CLUSTERFACTOR (F4) indicates REORG is necessary for indexes that are not in the same sequence as the base table. When multiple indexes are defined on a table, one or more indexes might be flagged as needing REORG. Specify the most important index for REORG sequencing.

Tables defined using the ORGANIZE BY clause and the corresponding dimension indexes have a '*' suffix to their names. The cardinality of a dimension index is equal to the active blocks statistic of the table.

9

LDAP searches and slow operations

In this chapter, we discuss how to improve LDAP searches and also how to identify operations that are slow.

9.1 Improving LDAP searches

Some LDAP searches are inherently slow and expensive. Consider:

```
ldapsearch -b o=everything cn=*smith*
```

DB2 does not do full text indexing. As a result, there is no way to resolve this search without doing a table scan for the CN table, which is very slow. This is not true for single wildcards such as cn=*smith or cn=smith*. Double wildcard searches are slow. Consider:

```
ldapsearch -b o=everything (&(!(cn=smith))(!(deleted=*)))
```

Searches with *NOT* clauses are slow. Recent fix packs for IBM Tivoli Directory Server include a fix for many such searches by substituting an SQL *NOT EXISTS* clause for the SQL *NOT* clause. However, it does not fix compound *NOT* statements such as the one above. Consider:

```
ldapsearch -b o=everything objectclass=person
```

This search might retrieve millions of entries. Searches with very large result sets are always slow.

Another search shows a related issue:

```
ldapsearch -b o=everything
(|(objectclass=widgets)(|(objectclass=gadgets)(objectclass=whozits)))
```

This search has the unfortunate characteristic that it references the object class table three times. References to the object class table should be avoided whenever possible. Assume that you can identify the same result set with the following search:

```
ldapsearch -b o=everything
(|(partType=widget)(|(partType=gadget)(partType=whozit)))
```

This search is much faster (provided that partType is indexed) because the partType table, like the table for every attribute type, is much smaller than the object class table. The search clause (objectclass=*) is free of cost. The directory recognizes that every entry satisfies this filter, therefore it generates no SQL for this clause. The search clause (objectclass=top) is expensive, although it returns the same result set. Tivoli Directory Server generates SQL for this clause, which is expensive to evaluate.

Simple searches that contain few Boolean clauses are always faster than complex searches. Therefore, it is always best to use the simplest search that provides a given result set. However, there is no performance advantage to be gained by reordering the clauses in a search filter. DB2 chooses the best order automatically.

Every attribute type that is used in a search filter should be indexed. If the table is not indexed, the search will force a table scan, which is always bad in large directories.

In some cases, a complex and slow search can be replaced by a simple search that returns a slightly larger result set. The extra entries can be filtered out on the application side. Consider the search:

```
ldapsearch -b ou=verySmall, o=everything (&(!(deleted=*))(!(suspended=*)))
```

If very few entries have values for the *deleted* and *suspended* attributes, then it is much faster to filter them out on the application side.

Tivoli Directory Server supports alias dereferencing so that entries in the directory might be known by several names. Alias dereferencing is always slow. Aliases should not be used in large directories, and alias dereferencing should be turned off in the directory.

9.2 Identifying slow operations

It can be helpful to recognize which operations are slow. In some cases, it might be possible to change the applications or workload to reduce slow operations. In other cases, the directory can be tuned to speed up these operations, for example, by indexing an attribute type used in a search filter.

To identify slow operations, you should run the audit log, logging all operations. The audit log records data for each operation, with timestamps for both the request and response. By comparing the timestamps, you can identify the particular operations that are slow. Example 9-1 shows a sample of audit log data with response times.

Example 9-1 Audit log data

```
AuditV2--2005-09-15-05:50:33.399-06:00DST--V3 Search--bindDN: cn=Directory
Manager--client: 10.33.20.33:21496--connectionID: 17--received:
2005-09-15-05:50:33.398-06:00DST--Success
base: o=acme
scope: singleLevel
derefAliases: derefAlways
typesOnly: false
filter: (objectclass=ERTENANT)
```

This search is almost instantaneous, at roughly a millisecond. However the following search, shown in Example 9-2, takes almost 7 seconds.

Example 9-2 Audit log data

```
AuditV2--2005-09-15-07:21:55.052-06:00DST--V3 Search--bindDN: cn=Directory
Manager--client: 10.33.20.34:1411--connectionID: 2--received:
2005-09-15-07:21:48.135-06:00DST--Success
base: ou=acme, o=acme
scope: wholeSubtree
derefAliases: derefAlways
typesOnly: false
filter: (&(!(erisdeleted=Y))(namingcontexts=DC=HRLOAD)(objectclass=ERSERVICEITEM))
```

Any search taking over 100 milliseconds is a problem search and should be corrected. In many cases, appropriate tuning, such as a change to the indexes, will correct the problem. In a few cases, a change within the application might be required (the 7-second search above is corrected by the latest IBM Tivoli Directory Server fix pack, which improves the SQL for searches with *NOT* clauses).

10

Indexes and direct I/O

In this chapter, we provide an overview of indexes and direct I/O.

10.1 Indexes explained

Attributes within the directory can be indexed. In general, if an attribute is used frequently in a search filter, it should be indexed. Some attributes, such as *UID* and *object class*, are always indexed. Different kinds of indexes can be created, depending on how the attribute type is used in the search:

myAttribute=*	Index the attribute type for Equality
myAttribute=blue	Index the attribute type for Equality
(!(myAttribute=blue))	Index the attribute type for Equality
myAttribute>=5 or	
myAttribute<=5	Index the attribute type for Ordering
myAttribute=start*	Index the attribute type for Substring
myAttribute=*end	Index the attribute type for Reverse
myAttribute~=blew	Index the attribute type for Approximate The index that gets created for Approximate can be very large. Approximate searches are very rarely used in Lightweight Directory Access Protocol (LDAP), therefore this index is rarely needed.

For any attribute type, you can specify a combination of these indexes. Sometimes a sort control is specified so that entries can be returned in sorted order. Each attribute that is used in a sort control should be indexed for ordering. Binary attributes, such as jpeg photographs, cannot be used in search filters and cannot be indexed.

Do not specify indexes for attributes unless those attributes are used in search filters or sort controls. The additional indexes add to the size of the directory and slow adds, modifies, and deletes for the corresponding entries. Instructions for creating or deleting indexes are provided in the "Working with attributes" section of the *IBM Tivoli Directory Server Administration Guide Version 6.0*, SC32-1674.

10.1.1 Optimizing indexes using DB2 commands

DB2 indexing for LDAP is evolving. We have learned several ways to improve the indexes that are created. The LDAP directory automatically creates indexes as follows:

```
db2 CREATE INDEX "LDAPDB2 "."LDAP_DESC_AEID" ON "LDAPDB2 "."LDAP_DESC" ("DEID"
ASC,"AEID" ASC)
```

We have learned that it is better to create the indexes with ALLOW REVERSE SCANS and MINPCTUSED 10 as follows:

```
db2 connect to ldapdb2
db2 drop INDEX "LDAPDB2 "."LDAP_DESC_DEID"
db2 commit
db2 CREATE INDEX "LDAPDB2 "."LDAP_DESC_DEID" ON "LDAPDB2 "."LDAP_DESC" ("AEID"
ASC,"DEID" ASC) MINPCTUSED 10 ALLOW REVERSE SCANS
db2 commit
```

The option ALLOW REVERSE SCANS can speed search times. The option MINPCTUSED 10 dynamically reorganizes the index so that sparsely filled consecutive pages are combined. This minimizes the need to do reorgs on the index and keeps performance good, as directory updates progress. This optimization can be made for all the LDAPDB2 indexes. You can list all of them by name using the statement:

```
db2 connect to ldapdb2
db2 select TABNAME,INDNAME,COLNAMES,UNIQUERULE where USER_DEFINED=1
```

Note: By applying Tivoli Directory Server 6 Fix Pack 2 before you configure and load your directory, the following options are added to all table and indexes that are created from that time onwards: ALLOW REVERSE SCANS and MINPCTUSED 10.

For a few indexes, it might also make sense to set the uniqueness rule as follows:

```
CREATE UNIQUE INDEX LDAPDB2.LDAP_ENTRY_PEID2 ON LDAPDB2.LDAP_ENTRY (PEID ASC, EID
ASC) PCTFREE 10 MINPCTUSED 10 ALLOW REVERSE SCANS;
```

The uniqueness rule must be used with care. For example, if you want to find all entries with a given attribute value and there are multiple entries with the same attribute value, this rule must not be used, because the resulting index finds only one of the entries. Each entry has a unique parent ID, however, therefore setting uniqueness on the parent ID index makes sense.

Do *not* change these indexes, unless you know what you are doing.

10.1.2 Optimizing searches using DB2 explain

Assume that the audit log shows a long search time. Assume further that each of the attributes referenced in the search filter is indexed. It might be difficult to determine why the search is slow, but DB2 provides two useful facilities, the DB2 snapshot and the DB2 explain commands.

To get a useful snapshot, you can use the following series of commands. Here it is assumed that ldapdb2 is the name of the DB2 instance and database name. This snapshot must be taken while the system is under a full test load, so as to catch the problems as they occur.

```
su - ldapdb2
db2 connect to ldapdb2
db2 update monitor switches using bufferpool on sort on table on
statement on uow on lock on
db2 reset monitor all
< wait 5 minutes >
db2 get snapshot for all on ldapdb2 > snap.out
db2 connect reset
```

Be aware that the resulting file can be large. In earlier sections, we have described some of the trouble spots to look for, such as dirty page steal or sort buffer overflows. This report might also identify particular troublesome statements. A sample output is shown in Example 10-1.

Example 10-1 DB2 monitor output

```
Number of executions                = 163
Number of compilations              = 1
Worst preparation time (ms)         = 4
Best preparation time (ms)          = 2
Internal rows deleted               = 0
Internal rows inserted              = 0
```

```
Rows read                          = 0
Internal rows updated              = 0
Rows written                       = 0
Statement sorts                    = 0
Statement sort overflows           = 0
Total sort time                    = 0
Buffer pool data logical reads     = 0
Buffer pool data physical reads    = 0
Buffer pool temporary data logical reads   = 0
Buffer pool temporary data physical reads  = 0
Buffer pool index logical reads    = 1660956145
Buffer pool index physical reads   = 19
Buffer pool temporary index logical reads  = 0
Buffer pool temporary index physical reads = 0
Total execution time (sec.ms)      = 14703.064392
Total user cpu time (sec.ms)       = 2673.319051
Total system cpu time (sec.ms)     = 7.429755
Statement text                     = SELECT distinct D.DEID FROM LDAPDB2.LDAP_DESC AS D WHERE
D.AEID=? AND D.DEID IN  (SELECT EID FROM LDAPDB2.UID)  FOR FETCH ONLY
```

This search shows a total execution time of over 14000 seconds for a statement that is executed only 163 times. The average execution time is 90 seconds for each search. This is very high. Relatively few physical reads are occurring, but lots of buffer pool index logical reads are occurring. DB2 is reading a lot of index data in memory, and this is causing high CPU consumption and long search times. The snapshot shows the SQL for the troublesome search, but we can figure out what the original LDAP search was.

We see that LDAP_DESC is used in the SQL. That means it is a subtree search. We see that the only other table referenced in the search is the UID table. That means that UID is the only attribute in the search filter. Therefore, the slow search is of the form:

```
ldapsearch -b <base DN> -s subtree uid=?
```

The snapshot does not show the specific values of base DN or UID used in the search. This statement might have been executed many times with different base DN and UID values. However, we can look in the audit log to see if there is a slow search of this form, and we might see an example such as:

```
ldapsearch -b "o=Acme, c=us" -s subtree uid=FSKEY
```

You can try this search from the command line to duplicate the problem. We now use the explain statement to understand what might be causing the problem. There are several steps.

1. We create the explain tables as follows.

   ```
   su - ldapdb2
   db2 connect to ldapdb2
   db2 -tvf sqllib/misc/EXPLAIN.DDL
   ```

2. We create a file, db2sql.txt, with the troublesome SQL.

   ```
   SELECT distinct D.DEID FROM DBLDP1Z.LDAP_DESC AS D WHERE D.AEID=? AND D.DEID IN
   (SELECT EID FROM LDAPDB2.UID)  FOR FETCH ONLY;
   ```

3. We get the explain data, and format the output in a file.

   ```
   db2advis -d ldapdb2 -p -i db2sql.txt
   db2exfmt -d <dbname> -o <outfile>
   ```

The resulting file might look similar to that shown in Example 10-2.

Example 10-2 DB2 example output

```
DB2 Universal Database Version 8.1, 5622-044 (c) Copyright IBM Corp. 1991, 2002
Licensed Material - Program Property of IBM
IBM DB2 Universal Database SQL Explain Tool

******************** DYNAMIC ***************************************

==================== STATEMENT ==========================================

   Isolation Level          = Cursor Stability
   Blocking                 = Block Unambiguous Cursors
   Query Optimization Class = 5

   Partition Parallel       = No
   Intra-Partition Parallel = No

   SQL Path                 = "SYSIBM", "SYSFUN", "SYSPROC", "LDAPDB2"


SQL Statement:

  select distinct d.deid
  from dbldp1z.ldap_desc as d
  where d.aeid=133333 and d.deid in (
     select d.deid
     from ldapdb2.uid)
  for
  fetch only


Section Code Page = 1208

Estimated Cost = 2832063.750000
Estimated Cardinality = 1088279.000000

Access Table Name = LDAPDB2.LDAP_DESC  ID = 3,4
|  Index Scan:  Name = LDADB2.LDAP_DESC_DEID  ID = 2
|  |  Regular Index (Not Clustered)
|  |  Index Columns:
|  |  |  1: AEID (Ascending)
|  |  |  2: DEID (Ascending)
|  #Columns = 1
|  #Key Columns = 2
|  |  Start Key: Inclusive Value
|  |  |  |  1: 133333
|  |  Stop Key: Exclusive Value
|  |  |  |  1: 133333
|  |  |  |  2: NULL
|  Index-Only Access
|  Index Prefetch: None
|  Lock Intents
|  |  Table: Intent Share
|  |  Row  : Next Key Share
```

```
Nested Loop Join
|  Access Table Name = LDAPDB2.UID  ID = 3,791
|  |  Index Scan:  Name = LDAPDB2.UIDI  ID = 1
|  |  |  Regular Index (Not Clustered)
|  |  |  Index Columns:
|  |  |  |  1: EID (Ascending)
|  |  #Columns = 0
|  |  Single Record
|  |  #Key Columns = 0
|  |  |  Start Key: Beginning of Index
|  |  |  Stop Key: End of Index
|  |  Index-Only Access
|  |  Index Prefetch: Eligible 9942
|  |  Lock Intents
|  |  |  Table: Intent Share
|  |  |  Row  : Next Key Share
Distinct Filter  #Columns = 1
Return Data to Application
|  #Columns = 1

End of section


Optimizer Plan:

                       RETURN
                       (   1)
                          |
                       UNIQUE
                       (   2)
                          |
                       NLJOIN
                       (   3)
        /-----/                    \-----\
        IXSCAN                        IXSCAN
        (   4)                        (   3)
        /      \                     /      \
 Index:          Table:          Index:       Table:
 LDAPDB2         LDAPDB2        LDAPDB2      LDAPDB2
LDAP_DESC_DEID   LDAP_DESC       UIDI         UID
```

When executing the query plan, DB2 starts at the bottom left. Therefore, it finds all descendants of the given search base DN. Then it checks each one (NLJOIN stands for *nested loop join*) to see if the UID matches. This is a very poor way to perform this search, because there are many entries under the specified base DN. A much better way is to find the matching UID first, and then to check the descendant table to make sure that the returned entry is a descendant of the base DN. How do we make it do this? The first step is to run **runstats** on both of the tables. This might correct the query plan, because the cardinality of the UID table should be much smaller than the cardinality of the descendant table. We also see a curiosity here: LDAP_DESC_DEID has been defined, but it has been defined with columns AEID, DEID. It should be defined with columns DEID, AEID. Therefore, we fix the index and run **runstats** once more. Then we check to see if the response time for this search is fast, and we get the explain data again if it is not.

If these steps do not suffice to fix the query plan, we might want to set LDAP_MAXCARD ON. This causes the query optimizer to defer referencing LDAP_DESC until the last step, therefore it will definitely fix the query plan for this search. But LDAP_MAXCARD can have side effects for other searches, therefore we have to test the effect on all searches to see if the net effect on the given workload is positive.

10.2 Direct I/O

By default, on all operating systems for which Tivoli Directory Server is available, reads and writes are buffered by the file system. An assumption of file system design is that, the same block is often read several times in a given time interval so that file system caching reduces the number of physical disk reads. However, file system buffering does not help DB2 reads from the tablespaces. All DB2 reads and writes to tablespaces are handled as memory mapped I/O within DB2, and are buffered in the DB2 buffer pools. This makes file system buffering redundant, therefore it should be turned off. There are several ways to do this. If raw database-managed storage (DMS) containers for the tablespaces are used, I/O is always direct to disk. However, DMS containers should not be used for large directories because, within a DMS container, no table can be over 64 GB. A directory with millions of entries might have tables that are larger than this. System-managed storage (SMS) containers should always be used for very large directories. Most of the advantages of DMS containers and direct I/O can be realized by disabling file system buffering as follows.

```
db2 alter tablespace userspace1  NO FILE SYSTEM CACHING
db2 alter tablespace LDAPSPACE  NO FILE SYSTEM CACHING
```

These commands should be executed before starting the server. The principal advantage is that CPU consumed in reads is reduced by roughly 30%. There is little effect on write loads. If *file system caching* was not turned off when the tablespace was created, you can do this by running the db2_tunings.sh when pre-tuning the DB2 database, when installing the Lightweight Directory Access Protocol (LDAP) instance. If you created your DB2 instances by DMS and used different names for these tablespaces, you have to change the names so that they match in this script also.

For some operating systems, an additional option is available to boost disk writes. AIX with JFS2 supports concurrent I/O. When the NO FILE SYSTEM CACHING option is used on a tablespace on an AIX JFS2 file system, concurrent I/O is enabled. Concurrent I/O can also be enabled on the level of the file system using the **`-cio`** option on the **`mount`** command.

```
mount -o cio <file system name>
```

This option uses direct I/O implicitly.

On Linux®, if you are using the 2.6 kernel or later, you can enable asynchronous I/O as follows:

1. Stop the directory.
2. Stop the database.
3. Issue this command.

   ```
   db2set DB2NOLIOAIO=false
   ```

The database and directory can then be restarted. In general, this approach is more complicated than using NO FILE SYSTEM CACHING, therefore NO FILE SYSTEM CACHING is preferable.

11

Disk striping and RAID

Using disk striping, the storage-array hardware can read and write to multiple disks simultaneously and independently. By allowing several read/write heads to work on the same task at once, disk striping can enhance performance. The amount of information read from or written to each disk makes up the stripe element size. The stripe size is the stripe element size multiplied by the number of disks in a group minus one. For example, assume that a stripe element size of 64 sectors and each sector is 512 bytes. If the group has five disks, the stripe size will be the element size, 64 sectors, times four disks, or 256 sectors, which equals 128 KB. The recommended disk striping configuration is shown in Figure 11-1.

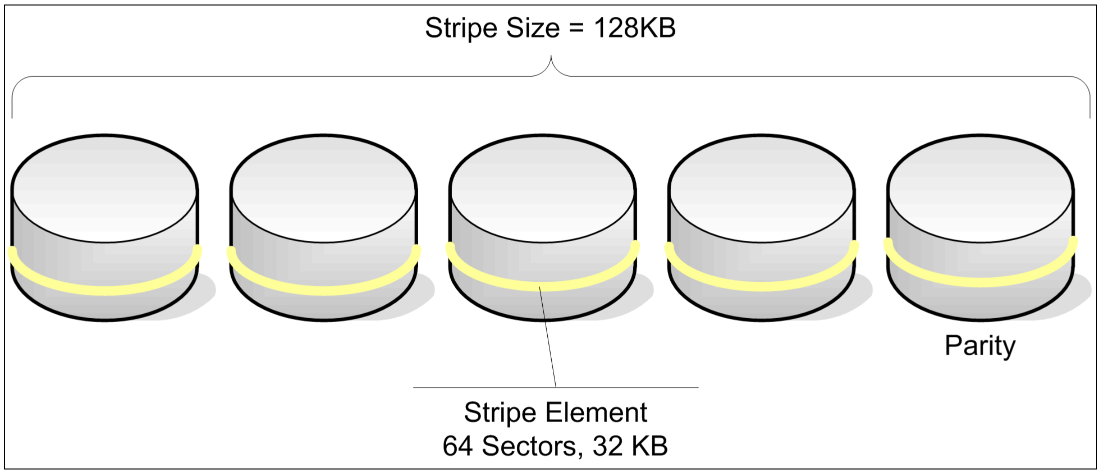


Figure 11-1 Disk striping

11.1 Considerations for RAID arrays

Large directories are frequently constrained by I/O speeds. This constraint can be relieved by spreading the I/O over multiple spindles, for example, in a Redundant Array of Independent Disks (RAID) array. RAID arrays have other advantages in that they provide very high availability and the ability to use volume snapshots and similar means for fast backup and restore of the database.

The best practices for use of RAID are described in the section “Optimizing tablespace performance when data is on RAID devices” in the “Planning” section of the *IBM DB2 Universal Database Administration Guide: Planning Version 8*, SC09-4822. Here are some key points:

- The DB2 transaction log should be on a different physical disk than the tablespaces.
- Either RAID 1 or RAID 5 can be used for tablespaces. RAID 1 should be used for the transaction log.
- A tablespace might span multiple physical disks, but it should be mounted as a single virtual disk. The RAID system provides the required parallelism. Multiple virtual disks (containers) for the tablespace slow the throughput by randomizing the I/O. DB2_PARALLEL_IO should be enabled.
- Make the EXTENTSIZE of the tablespace equal to, or a multiple of, the RAID stripe size. For LDAP, DB2 has a default extent size of 32 K.
- Ensure that the PREFETCHSIZE of the tablespace is:
  - The RAID stripe size multiplied by the number of RAID parallel devices (or a whole multiple of this product)

    And
  - A multiple of the EXTENTSIZE.

  For LDAP, DB2 has a default prefetch size of 32 K.
- Do not put several highly active databases on the same set of physical disks. Each busy database should have its own set of physical disks. For example, you should move the transaction logs to another mount point so that they are not on the same disk as the database. By default, they reside on the same disk space.

To give adequate performance and minimize disk access contention, it is often necessary to allocate more physical disks than is required just to store the data. The extra space can be used to store non-LDAP data that is less frequently accessed.

12

Buffer pool settings and sort buffer overflow

This chapter discusses buffer pool settings and sort heap threshold settings for very large directories.

© Copyright IBM Corp. 2007. All rights reserved.

12.1 Adjusting the buffer pool and sort heap threshold settings

For large directories of a million entries or more, it might be necessary to adjust the DB2 buffer pool settings, the sort heap threshold settings, or both these settings. A database snapshot gives the following information as shown in Table 12-1.

Table 12-1 Database snapshot information

Total sorts	1068692
Total sort time (ms)	1271408953
Sort overflows	15
LSN Gap cleaner triggers	11205
Dirty page steal cleaner triggers	99
Dirty page threshold cleaner triggers	0

Dirty page steal should be low in a Tivoli Directory Server deployment, by comparison with other forms of buffer pool cleaning activity. This is very important. If dirty page steal is high, that is, if more than 5% of the page cleaning triggers is for dirty page steal, then the buffer pool settings are not large enough. A larger value often means that temporary table allocation has spilled on to the disk, creating a lot of write activity, which is bad. *Sort overflows* are also bad and should be very infrequent. If they occur in more than 0.1% of the sorts, the sort heap threshold should be increased.

At the same time, it is important not to set these memory values too high. If the total memory allocated is more than is available, DB2 will arbitrarily set a very low value for the size of the buffer pools and bad performance will result due to not enough real memory to load the buffer pools.

You can use the db2_tunings.sh, which is included in the attached scripts, to perform the above steps. It takes 50% of your real memory and sets it up for DB2 buffer pools and auto sets up your ioservers and iocleaners to make use of this increase of memory for buffer pools. This script also sets up your sort heap size along with other important settings. See 3.2, "db2_tunings.sh" on page 17, for more information. These settings can be raised or lowered as needed.

13

Replicas and partitions for performance

Some Lightweight Directory Access Protocol (LDAP) directories have very high read and write rates—so high, in fact, that all of this tuning advice will not suffice to give the required level of throughput for a single database instance. The next step is to load balance read traffic across multiple replicas. Every Tivoli Directory Server deployment should include multiple replicas for high availability, so that if one fails, a backup can assume the load. If a single instance does not suffice to support the required level of read traffic, the read load can be balanced across multiple instances. Most Ethernet switches now contain a virtual IP function that can either load balance traffic between servers or fail over traffic from one server to another. The best policy is to load balance reads and fail over writes. You do not want to load balance writes because:

- No performance advantage will result as each write must be replicated to every peer master
- Load balancing writes creates the possibility of update conflicts, where a single entry is changed on several peer masters concurrently.

© Copyright IBM Corp. 2007. All rights reserved.

13.1 Distinguishing between LDAP reads and writes

Switches do not include logic to distinguish LDAP reads from writes. There are several ways to work around this problem. First, you can point all read-only applications to a load balancing virtual IP address, and all read-write applications to a virtual IP address configured for failover and not load balancing. Second, you can divide load so that the same entries are not written on different servers concurrently. For example, you can point all read and write activity for West Coast entries to one peer master, and all read and write activity for East Coast entries to another peer, where each fail over to the other. Third, you can use the Tivoli Directory Server directory proxy, which can distinguish between LDAP reads and writes and can load balance reads while doing failover for writes.

With Tivoli Directory Server Version 6 and later, replication can now be multi-threaded for any suffix except cn=ibmpolicies, which can only be single threaded. This is not an issue because cn=ibmpolicies suffix is only used for schema and password policies updates and does not have a high usage rate to warrant the need for multi-threading. Multi-threading can be turned on or off for each suffix per server. By default, all suffix replication is single threaded. If you find that one of your suffixes is taking too long to replicate and you have already looked at any network bottlenecks, then turning on multi-threading for that suffix might be what you have to do. Remember that each server in the replication agreement for that suffix has to be configured to use multi-threading to get the best results from turning on multi-threading.

Replication gives good scaling for read traffic, but it does not help to scale writes. Every replica contains all entries, therefore every write goes to each replica. A single instance can support several hundred of writes a second. Your mileage varies depending on the size of the write, the number of indexes, the amount of memory, the speed of the machine, and the speed and type of access to the storage disk and last but not the least, the number of entries in your database. The key to this is, if you are going to have high write rates along with high tens of millions of entries in your database, you must partition the directory. A directory is divided into two or more partitions, each holding a fraction of the data and handling a fraction of the writes. The Tivoli Directory Server proxy is used to route traffic to the appropriate partition. All writes and base searches can be routed to a single partition. Most subtree searches span multiple partitions, but the proxy sends search requests to servers for each partition and accumulates the results. One main requirement that goes with using partitions is that, when you decide how many partitions to use and load the data into each partition, you cannot just add another partition later on without dumping all the data and reloading it all across all the partitions again. This is due to the fact that decisions on what data goes on what partition is determined by a number of factors such as the number of partitions that will be used and the hash process being used. A lot of planning has to be put in place to design an enclave that will be using partitions.

A few applications, such as the Tivoli Identity Manager, require high data consistency. LDAP has a *loose consistency* data model, which means that a write operation returns a completion response to the calling application before the update has been replicated to all replicas. If an entry is written and immediately read in a replicated environment with load balancing, the read might go to a different server than the one where the write occurred, and the update might not yet have been replicated to that server. Applications that require tight consistency cannot use LDAP load balancing for scaling. However, you can use partitioning to scale LDAP for these applications because the partitioned image retains tight consistency.

14

LDAP replication information

In this chapter, we discuss several tips and hints on the topic of replication. Because Lightweight Directory Access Protocol (LDAP) replication has become much more powerful in the las few years, we take some time introducing the different replication terms used throughout this chapter and paper. After that, we discuss more specific topics.

© Copyright IBM Corp. 2007. All rights reserved.

14.1 Defining replication terms

The following list describes some of the terms related to replication:

- Definition of replication
  - Replication is a technique used by directory servers to improve performance, availability, and reliability.
  - The replication process keeps the data in multiple directory servers synchronized.
- Common reasons for replication
  - Redundancy of information
  - Replicas back up the content of their supplier servers
  - Replicas can be used to maintain a failover capability
- Faster searches
  - Search requests can be spread among several different servers instead of a single server.
  - Replica servers can reside in another geographic location, avoiding network delays.
- Security and content filtering

  Replicas can contain subsets of the data in a supplier server.
- Relative Distinguished Name (RDN™)

  The first component of the distinguished name (DN). For example, if the entry's DN is cn=John Doe, ou=Test, o=IBM, c=US, the RDN is cn=John Doe.
- Distinguished Name (DN)

  The name that uniquely identifies an entry in a directory. A distinguished name is made up of attribute=value pairs, separated by commas. For example, the entry's DN is cn=John Doe, ou=Test, o=IBM, c=US.
- Master
  - A server that is writable (can be updated) for a given LDAP subtree.
  - It is considered a *supplier* if it sends data to any replicas.
- Replica
  - May or may not be writable
  - Can also be a master
  - Consumes data sent to it by one of more masters (suppliers)
  - Contains all or a subset of its suppliers data
- Peer server

  The term used for a master server when there are multiple masters for a given subtree. A peer server does not replicate changes sent to it from another peer server; it only replicates changes that are originally made on it.
- Forwarding server

  A read-only server that replicates all changes sent to it. This contrasts with a peer/master server in that it is read-only and it can have no peers.
- Gateway server

  A server that forwards all replication traffic from the local replication site where it resides to other gateway servers in the replicating network. Also receives replication traffic from other gateway servers within the replication network, which it forwards to all servers on its local replication site. Gateway servers must be masters (writable).

- Consumer

  A server that receives changes through replication from another (supplier) server.

- Supplier

  A server that sends changes to another (consumer) server.

- Replication context
  - Identifies the root of a replicated subtree.

    The *ibm-replicationContext* auxiliary object class can be added to an entry to mark it as the root of a replicated area. The configuration information related to replication is maintained in a set of entries created below the base of a replication context.
  - Example LDAP Data Interchange Format (LDIF):

```
dn: o=IBM, c=US
objectclass: top
objectclass: organization
objectclass: ibm-replicationContext
o: IBM
ibm-replicareferralurl: ldap://localhost:389
```

- Replica group
  - The first entry created under a replication context has object class ibm-replicaGroup and represents a collection of servers participating in replication. It provides a convenient location to set access control lists (ACLs) to protect the replication topology information. The administration tools currently support one replica group under each replication context, named ibm-replicaGroup=default.
  - Example LDIF:

```
dn: ibm-replicaGroup=default, o=ibm, c=us
ibm-replicaGroup: default
objectclass: ibm-replicaGroup
objectclass: top
```

- Replica subentry
  - Below a replica group entry, one or more entries with object class ibm-replicaSubentry can be created; one for each server participating in replication as a supplier. The replica subentry identifies the role the server plays in replication: master or read-only. A read-only server might, in turn, have replication agreements to support cascading replication. One procedure that we talk later in this document about is the changing of the ibm-replicaServerId to a more friendly descriptive name to improve troubleshooting of issues that might show up in replication. For example, use the short name of the server and tag on -uid at the end (server20-uid). This replaces the 31 place unique hex number.
  - Example LDIF:

```
dn: cn=localhost:389, ibm-replicaGroup=default, o=ibm, c=us
objectclass: ibm-replicaSubentry
objectclass: top
ibm-replicaServerId: 4dbdc73a-2476-4039-8808-9655f95d917
ibm-replicationServerIsMaster: TRUE
cn: localhost:389
```

- Replication agreement
  - Information contained in the directory that defines the *connection* or *replication path* between two servers. One server is called the supplier (the one that sends the changes) and the other is the consumer (the one that receives the changes). The agreement contains all the information required for making a connection from the supplier to the consumer and scheduling replication.
  - Example LDIF

```
dn: cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid,
ibm-replicaGroup=default, O=IBM, C=US
objectclass: top
objectclass: ibm-replicationAgreement
cn: ldapserv3
ibm-replicaConsumerId: ldapserv3-uid
ibm-replicaUrl: ldap://ldapserv3:389
ibm-replicaCredentialsDN: cn= mycreds, cn=replication, cn=IBMpolicies
description: ldapserv1 (site1 peer master 1) to ldapserv3 (the site1
forwarder) agreement
```

- Replica credentials
  - Identifies the method and required information that the supplier uses in binding to the consumer. For simple binds, this includes the DN and password. The credentials are stored in an entry, the DN of which is specified in the replica agreement.
  - Example LDIF

```
dn: cn=mycreds, cn=replication, cn=IBMpolicies
replicacredentials: master
objectclass: ibm-replicationcredentials
objectclass: ibm-replicationcredentialssimple
objectclass: top
replicabinddn: cn=master
cn: mycreds
```

14.2 cn=ibmpolicies replication problem

The design of Tivoli Directory Server v6.0 is to have the ibm-replicationcontext, ibm-replicagroup, and the ibm-replicasubentry set up automatically for the cn=ibmpolicies subtree the first time that the LDAP server is started. In some circumstances, this has caused problems. It is recommended that you remove these replication entries from all machines in the topology before setting up the replication agreement for cn=ibmpolicies.

To remove the entries:

1. You have to search the LDAP servers to get the entries as shown in Example 14-1.

Example 14-1 Ldapsearch

```
==> ldapsearch -D cn=root -w secret -L -b cn=ibmpolicies
objectclass=ibm-replica*
dn: CN=IBMPOLICIES
cn: IBMpolicies
objectclass: container
objectclass: top
objectclass: ibm-replicationcontext
```

```
dn: ibm-replicagroup=default, cn=ibmpolicies
objectclass: top
objectclass: ibm-replicagroup
ibm-replicaGroup: default

dn: ibm-replicaserverid=ac1156c0-a214-1029-934c-cd9424fd6984,ibm-replicagroup=
default, cn=ibmpolicies
objectclass: top
objectclass: ibm-replicasubentry
ibm-replicationserverismaster: TRUE
cn: V6.0 Migration
ibm-replicaServerId: ac1156c0-a214-1029-934c-cd9424fd6984
```

Note: ac1156c0-a214-1029-934c-cd9424fd6984 is a randomly generated server-id; yours will be different.

2. Delete the ibm-replicasubentry.

```
ldapdelete -D cn=root -w ?
ibm-replicaserverid=ac1156c0-a214-1029-934c-cd9424fd6984,ibm-replicagroup=defau
lt,cn=ibmpolicies
```

3. Delete the ibm-replicagroup.

```
ldapdelete -D cn=root -w ?
ibm-replicagroup=default, cn=ibmpolicies
```

Now you should set up replication on the cn=ibmpolicies subtree just as you would any other subtree. This is discussed later on in this chapter.

14.3 Conflict resolution

There are several reasons for conflict resolution in Tivoli Directory Server.

- Tivoli Directory Server 4.1 introduces *peer to peer configuration*:
  - Originally used for a failover configuration
  - Tivoli Directory Server 4.1 only had complete server replication
  - Multiple masters can accept updates
- Tivoli Directory Server introduces subtree replication:
  - Individual masters can be primary server for a subtree of data.
  - Still used in failover configurations.
- Prior to Tivoli Directory Server 6.0, if multiple clients update the same data on different peers, the data might become inconsistent. This was not detected until a data error occurred or replication became blocked. A better solution was required.

Delete and rename operations

For delete and rename operations:

- No conflict resolution is provided yet.
- Example:
  - Peer1 gets a rename operation on a directory entry.
  - Peer2 gets a delete operation on the same directory entry.

  – If Peer1's operation goes to Replica1 first, then Peer2's delete will be trying to delete the object with the OLD name and fail.
  – If Peer2's operation goes to Replica1 first, then Peer1's rename will fail because the object will no longer exist.
- The administrator is required to rectify the data manually.

Add and modify operation

For add and modify operations:

- Conflicts are resolved at the consumer.
- Conflicts are resolved based on operation timestamp.
- It is *very* important to have all the servers' clocks in sync.
- The *most recent* operation takes precedence.
- The replaced entry is logged in the *Lost and Found* log. See Example 14-2 for more details.

Example 14-2 Lost and found log

```
version: 1
dn: cn=crypto, cn=localhost
cn: crypto
objectclass: ibm-cryptoConfig
objectclass: ibm-slapdConfigEntry
objectclass: top
ibm-slapdCryptoSync: YFILQ8pOKPsYBVmZ2gPe
ibm-slapdCryptoSalt: G]bpgi/vr[5:
ibm-entryuuid: b14b6ab4-669d-4e1d-b0ea-13d7978f0ea9

#Entry DN: ou=Austin, o=IBM, c=US
#Operation type:Add
#Corrective action:Replace
#Entry createTimestamp: 20051118164246.000000Z
#Entry modifyTimestamp: 20051205214351.000000Z
#Supplier address: 127.0.0.1
dn: ou=Austin, o=IBM, c=US
ou: Austin
objectclass: top
objectclass: organizationalUnit
seealso: cn=Linda Carlesberg, ou=Austin, o=IBM, c=US
description: changed on 389
ibm-entryuuid: 564fb7fe-404f-4c3a-8ba4-3d9218841679
```

Note: Regarding conflict resolution with more than two peer-to-peer servers or when you have a forwarder replica: When two or more writes to the same LDAP dn at the same time are executed, a conflict will occur.

With Tivoli Directory Server v6, replication now uses timestamps (new in Tivoli Directory Server v6) to keep track of all changes made and the conflict resolution process tries and resolves which change will take place. This process has overhead along with a number of retries (10) before it discards a conflict.

When using Access Manager for e-business or any other application that directs all write operations to only one master at any one time, or that is using a failover type load balancer between the application and Tivoli Directory Server v6, you can turn this process off and save a lot of overhead. If, for any reason, you use an application that sends the write operation to two or more LDAP peer masters, then with this environment variable set, the server does not try to compare the entries' timestamps for replicated entries in an attempt to resolve conflicts between the entries. What happens is, it executes each request one at a time and makes changes in the order they were received.

With *conflict resolution* turned on and your Tivoli Directory Server v6 servers reporting high CPU load and lots of entries in the lostandfound.log on each of the peer or forwarder servers, it might be a good idea to review the technote *Disabling replication conflict resolution* at the following location:

http://www.ibm.com/support/docview.wss?rs=767&context=SSVJJU&q1=conflict+resolution&uid=swg21236775&loc=en_US&cs=utf-8&lang=en

When the IBMSLAPD_REPL_NO_CONFLICT_RESOLUTION environment variable is defined, no conflict resolution takes place on a server. If the variable is defined before a server is started, the server operates in a *no replication conflict resolution* mode. This environment variable is checked during server startup, and therefore changing it while the server is running does not have any effect on the server.

We recommend that you turn off conflict resolution in your LDAP environment when you have more then two peers or forwarder replicas in your network and you are using Tivoli Access Manager for e-business or any other product that only writes to one LDAP server and fail over to the other peer master servers if one peer master server is not working.

To turn off conflict resolution, add the following to the cn=Front End, cn=Configuration section of the ibmslapd.conf file along with any other ibm-slapdSetenv variable that might be there:

```
"ibm-slapdSetenv: IBMSLAPD_REPL_NO_CONFLICT_RESOLUTION=true"
```

This has to be done on each LDAP server in the enclave. A restart of the ibmslapd process must be done to have this take affect. See Example 14-3 for details.

Example 14-3 ibmslapd.conf excerpt

```
dn: cn=Front End, cn=Configuration
cn: Front End
ibm-slapdACLCache: TRUE
ibm-slapdACLCacheSize: 25000
ibm-slapdEntryCacheSize: 25000
ibm-slapdFilterCacheBypassLimit: 100
ibm-slapdFilterCacheSize: 25000
ibm-slapdIdleTimeOut: 300
ibm-slapdSetenv: DB2CODEPAGE=1208
ibm-slapdSetenv: LDAP_MAXCARD=YES
ibm-slapdSetenv: IBMSLAPD_REPL_NO_CONFLICT_RESOLUTION=true
```

14.4 Monitoring and managing replication

Tivoli Directory Server provides operational attributes and extended operations to aid in monitoring, managing, and debugging replication.

14.4.1 Operational attributes

The following operational attributes can help administrators with monitoring for and debugging replication problems. They are also extensively used by the Web administration tool:

- ibm-replicationLastActivationTime

  This is attribute shows the time that the last replication session started between this supplier and consumer.

  ```
  C:\>ldapsearch -D cn=root -w secret -b
  cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid, ibm-replicaGroup=default,
  O=IBM, C=US objectclass=* ibm-replicationLastActivationTime
  cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid, ibm-replicaGroup=default,
  O=IBM, C=US
  ibm-replicationLastActivationTime=20051025182742Z
  ```

- ibm-replicationLastFinishTime

  This attribute shows the time that the last replication session finished between this supplier and consumer.

- ibm-replicationChangeId

  This attribute shows the changeID of the last update sent to this consumer.

  ```
  C:\>ldapsearch -D cn=root -w secret -b
  cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid, ibm-replicaGroup=default,
  O=IBM, C=US objectclass=* ibm-replicationLastChangeId
  cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid, ibm-replicaGroup=default,
  O=IBM, C=US
  ibm-replicationLastChangeId=11
  ```

- ibm-replicationState:

  This attribute is the current state of replication with this consumer.

Active	Actively sending updates to consumer
Ready	In immediate replication mode, ready to send updates as they occur
Waiting	Waiting for next scheduled replication time
Binding	In the process of binding to the consumer
Connecting	In the process of connecting to the consumer
On Hold	This replication agreement has been suspended
Error Log Full	For a server configured to use multiple connections, replication is suspended for this agreement. The receiver threads continue polling for status from any updates that have been sent, but no more updates are replicated.
Retrying	If the server is configured to use a single connection, replication attempts to send the same update after waiting for 60 seconds and keeps trying until replication succeeds or the administrator skips the update.

For the following examples, we have taken the replica cn=localhost:1389 offline, and added the following entry to create a pending change in the queue:

```
C:\>ldapadd -D cn=root -w secret -p 389
dn: o=test1, o=ibm, c=us
objectclass: organization
adding new entry o=test1, o=ibm, c=us
```

- ibm-replicationLastResult

  This attribute shows the results of the last attempted update to this consumer, in the form:

  ```
  <time stamp> <change id> <result code> <operation> <entry DN>
  C:\>ldapsearch -D cn=root -w secret -b
  cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid, ibm-replicaGroup=default,
  O=IBM, C=US objectclass=* ibm-replicationlastresult
  ibm-replicationlastresult=20051025215549Z 12 81 add o=test1, o=ibm, c=us
  ```

- ibm-replicationPendingChangeCount

  This attribute shows the number of updates queued to be replicated to this consumer.

  ```
  C:\>ldapsearch -D cn=root -w secret -b
  cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid, ibm-replicaGroup=default,
  O=IBM, C=US objectclass=* ibm-replicationpendingchangecount
  ibm-replicationpendingchangecount=1
  ```

- ibm-replicationPendingchanges

  Each value of this attribute gives information about one of the pending changes in the form:

  ```
  <change id> <operation> <entry DN>
  ```

  Requesting this attribute might return many values. Check the change count before requesting this attribute.

  ```
  C:\>ldapsearch -D cn=root -w secret -b
  cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid, ibm-replicaGroup=default,
  O=IBM, C=US objectclass=* ibm-replicationpendingchanges
     ibm-replicationpendingchanges=12 add o=test1, o=ibm, c=us
  ```

- ibm-replicationChangeLDIF

  This attribute gives the full details of the last failing update in LDIF.

  Note: Single-threaded replication only.

- ibm-replicationFailedChanges

  This attribute lists the IDs, DNs, update types, result codes, timestamps, and number of attempts for failures logged for a specified replication agreement. The number of failures displayed are less than or equal to ibmslapdMaxPendingChangesDisplayed.

- ibm-replicationFailedChangeCount

  This attribute returns a count of the failures logged for a specified replication agreement.

- ibm-replicationIsQuiesced

  This attribute returns Boolean value that indicates if the subtree has been quiesced. Base dn of search should be replicationContext.

14.4.2 Extended operations

The following extended operations can be used to manage replication.

- cascrepl

  This is the cascading control replication extended operation.

  ```
  C:\>ldapexop -op cascrepl -help
  ```

  The usage for cascaded replication operation is:

  ```
  -op cascrepl -action actionValue -rc contextDn [options]
  ```

  Where:

  – actionValue:

    • quiesce: Quiesce the context
    • unquiesce: Unquiesce the context
    • replnow: Start immediate replication
    • wait: Wait for pending changes to be replicated

  – contextDn: specifies the root of the subtree

  – options

    • timeout secs: Timeout period in seconds

  For example:

  ```
  idsldapexop -op cascrepl -action quiesce -rc "o=ibm, c=us"
  ```

- controlqueue

  This is the control queue extended operation.

  ```
  C:\>ldapexop -op controlque -help
  ```

  The usage for control replication queue operation is:

  ```
  -op controlqueue -skip skipValue -ra agreementDn
  ```

  Where:

  – skipValue

    • all: Skip all pending changes for this agreement
    • change-id: Skip the specified change

  – agreementDn: DN of the replication agreement

  For example:

  ```
  idsldapexop -op controlque -skip all -ra
  cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid, ibm-replicaGroup=default,
  O=IBM, C=US
  ```

- controlrepl

  This is the control replication extended operation.

  ```
  C:\>ldapexop -op controlrepl -help
  ```

  The usage for control replication operation is:

  ```
  -op controlrepl -action actionValue -ra agreementDn
  ```

  Or

  ```
  -op controlrepl -action actionValue -rc contextDn
  ```

Where:

- actionValue
  - suspend: Suspend replication
  - resume: Resume replication
  - replnow: Start immediate replication
- contextDn: Specifies the root of the replication context

  The action will be performed for all agreements for this context.
- agreementDn: Specifies the replication agreement

  The action will be performed for the specified agreement.

For example:

```
idsldapexop -op controlrepl -action suspend -ra
"cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid, ibm-replicaGroup=default,
O=IBM, C=US"
```

- controlreplerr

  This is the control replication error extended operation.

```
C:\>ldapexop -op controlreplerr -help
```

The usage for controlreplerr operation is:

```
-op controlreplerr -show failure_ID -ra agreementDn
```

Where:

- failure_ID: Specifies the ID of the target replication update failure
- agreementDn: Specifies the DN of the replication agreement

```
-op controlreplerr -delete failure_ID -ra agreementDn
-op controlreplerr -retry  failure_ID -ra agreementDn
```

Where:

- failure_ID: Specifies the ID of the target replication update failure or ‘0’ for all
- agreementDn: Specifies the DN of the replication agreement

For example:

```
idsldapexop -op controlreplerr -delete all -ra
"cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid, ibm-replicaGroup=default,
O=IBM, C=US"
```

- repltopology

  This is the replication topology extended operation. Replicates the replication topology related entries under the specified context.

```
C:\>ldapexop -op repltopology -help
```

The usage for repltopology operation is:

-op repltopology -rc contextDn [options]

Where:

- contextDn: specifies the root of the subtree

Options:

- -timeout secs: Timeout period in seconds
- -ra agreementDn: Specifies the replication agreement.Entries will be propagating to the specified agreement.

For example:

```
idsldapexop -op repltopology -rc o=ibm, c=us -ra
"cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid, ibm-replicaGroup=default,
O=IBM, C=US"
```

14.4.3 Troubleshooting replication problems

Let us look at some debugging practices.

- If multiple replication agreements are failing, it is best to get one supplier-consumer link working properly and then move to the next failure.
- The ibmslapd.log file is the best point to start troubleshooting.
- Start with the supplier's log file.
  - Check to make sure it can connect to the consumer.
  - Check whether it is able to bind correctly.
  - Check the reason why replication is failing.

Here are some examples of common replication errors.

-
  ```
  Oct 19 14:20:23 2005  GLPRPL036E Error Invalid credentials occurred for replica
  'cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid, ibm-replicaGroup=default,
  O=IBM, C=US': bind failed using masterDn 'cn=master'
  ```

  Credentials defined by the replication agreement do not match what is defined in the consumer's ibmslapd.conf file.

-
  ```
  Nov 03 12:18:43 2005  GLPRPL036E Error Can't contact LDAP server occurred for
  replica 'cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid,
  ibm-replicaGroup=default, O=IBM, C=US': bind failed using masterDn 'cn=master'.
  ```

  Consumer LDAP server is not reachable from the master. Make sure that the consumer is online. Try using the supplier's ldapsearch client to query the consumer.

-
  ```
  Nov 03 22:28:54 2005  GLPRPL032E Error No such object occurred for replica
  'cn=ldapserv3,ibm-replicaServerId=ldapserv1-uid, ibm-replicaGroup=default,
  O=IBM, C=US': add failed for entry 'cn=thomas, o=support employees, o=ibm, c=us'
  change ID 7.
  ```

  A parent entry does not exist on consumer. Check the consumer's ibmslapd.log for the exact error. Servers can get out of sync for different reasons: never sync'd, updates sent to more than one master, and so on. The problem can be corrected by adding the missing entry to the consumer. You should bind as the replicabinddn.

14.5 Introduction to forwarders and gateways

Let use take a closer look at the forwarders and gateways in our LDAP environment.

14.5.1 Forwarders

A peer/master server replicates to a set of read-only (forwarding) servers that in turn replicate to other replica servers. Cascading replication offloads replication work from the peer/master server.

The supplier-consumer relationship for this scenario, depicted in Figure 14-1, is:

- The master is a supplier to the forwarder.
- The forwarder has two roles:
  - A consumer of the master
  - A supplier to the replica
- The replica is a consumer of the forwarder.

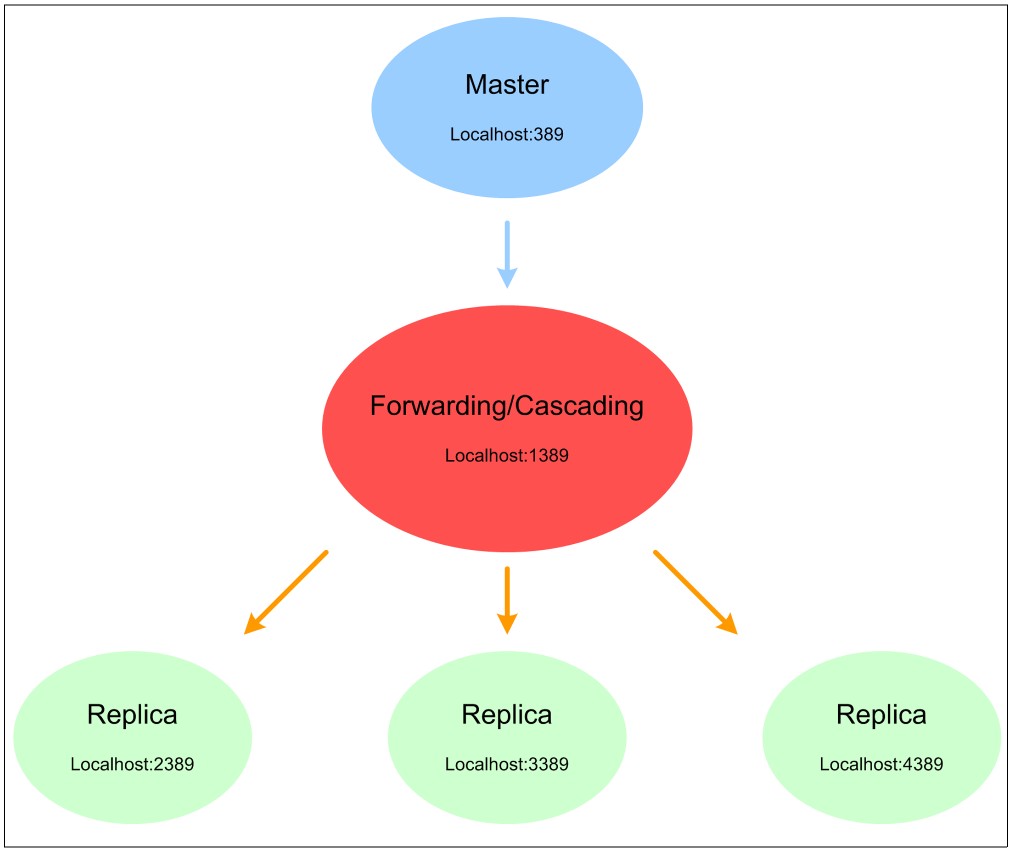


Figure 14-1 Overview of the forwarder functionality

Example 14-4 shows how to create replication credentials between the forwarder and its replicas.

Example 14-4 Replication credentials between the forwarder and its replicas

```
dn: cn=ldaptiv3 BindCredentials, cn=replication, cn=IBMpolicies
objectclass: ibm-replicationCredentialsSimple
cn: ldaptiv3 BindCredentials
replicaBindDN: cn=any
replicaCredentials: secret
description: Bindmethod of ldaptiv2 (the forwarder) to ldaptiv3 (the replica)

dn: ibm-replicaServerId=ldaptiv1-serverid ibm-replicaGroup=default, o=ibm, c=us
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: ldaptiv1-serverid
#true if master, false if forwarder
ibm-replicationServerIsMaster: true
```

```
cn: server1 description: master ibm-replicaSubentry
dn: ibm-replicaServerId=ldaptiv2-serverid ibm-replicaGroup=default, o=ibm, c=us
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: ldaptiv2-serverid
ibm-replicationServerIsMaster: false
cn: ldaptiv2
description: forwarder ibm-replicaSubentry

dn: cn=forwarder1,ibm-replicaServerId=ldaptiv1-serveridibm-replicaGroup=default,o=ibm,c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: ldaptiv2
ibm-replicaConsumerId: ldaptiv2-serverid
ibm-replicaUrl: ldap://ldaptiv2:389
ibm-replicaCredentialsDN: cn=ldaptiv2
BindCredentials, cn=replication, cn=IBMpolicies
description: ldaptiv1 (the master) to ldaptiv2 (the forwarder) agreement

dn: cn=ldaptiv3, ibm-replicaServerId=ldaptiv2-serverid ibm-replicaGroup=default, o=ibm, c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: ldaptiv3
ibm-replicaConsumerId: ldaptiv3-serverid-uid
ibm-replicaUrl: ldap://ldaptiv3:389
ibm-replicaCredentialsDN: cn=ldaptiv3
BindCredentials, cn=replication, cn=IBMpolicies
description: ldaptiv2 (the forwarder) to ldaptiv3 (the replica) agreement
```

14.5.2 Gateways

Gateway servers must be masters (writable). If you try and convert a read-only replica, you will get an error message. You must have at least two gateway servers between a replicating network. The primary benefit of a gateway replication is the *reduction of network traffic*.

Gateway replication updates from the peer/master servers in the replication site and sends the updates to all the other gateway servers within the replicating network. The replication process collects replication updates from other gateway servers in the replication network and sends those updates to the peers/masters and replicas in the replication site where it resides.

There are two methods for creating a gateway server. You can:

- Create a new gateway server
- Convert an existing peer server to a gateway server

See Figure 14-2 for more details.

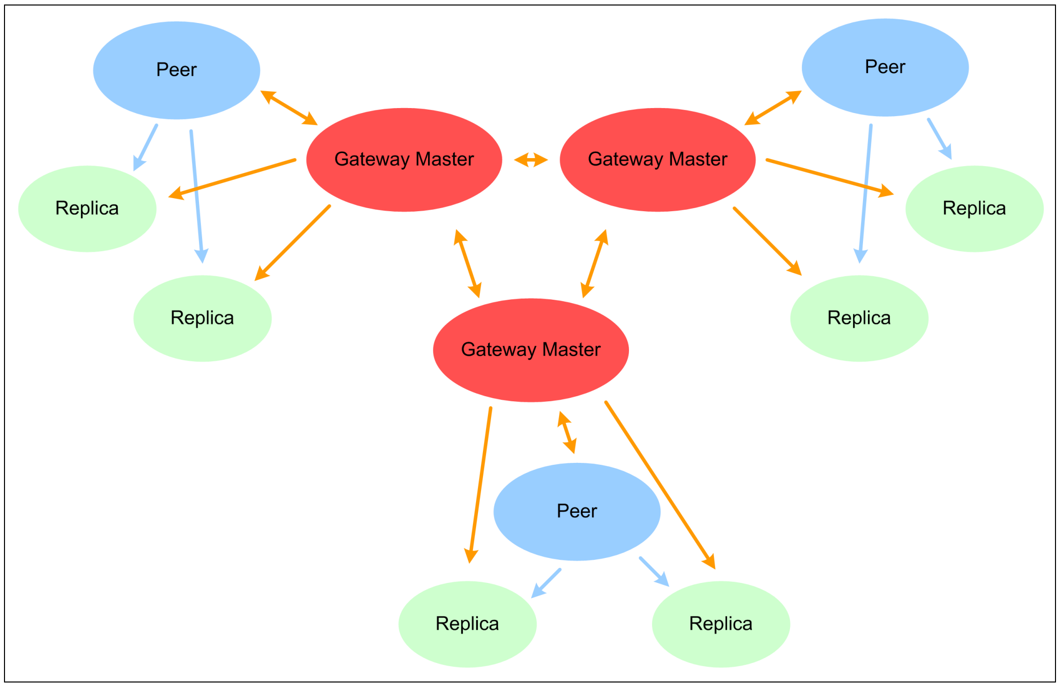


Figure 14-2 Overview of the gateway functionality

The replica subentry must contain the ibm-replicaSubentry object class and ibm-replicaGateway auxiliary object class along with ibm-replicationServerIsMaster: TRUE. See Example 14-5.

Example 14-5 Replication credentials between the gateway and its replicas

```
dn: ibm-replicaServerId=ldaptiv1, ibm-replicagroup=default, o=ibm, c=us
objectclass: top
objectclass: ibm-replicaSubentry
objectclass: ibm-replicaGateway
ibm-replicaServerId: ldaptiv1- serverid-uid
ibm-replicationServerIsMaster: TRUE
cn: ldaptiv1
description: server1 (gateway server from replication site 1 to replication site 2)

dn: ibm-replicaServerId=ldaptiv2, ibm-replicagroup=default, o=ibm, c=us
objectclass: top
objectclass: ibm-replicaSubentry
objectclass: ibm-replicaGateway
ibm-replicaServerId:ldaptiv2- serverid-uid
ibm-replicationServerIsMaster: TRUE
cn: ldaptiv2
description: server1 (gateway server from replication site 2 to replication site 1)
dn: cn=ldaptiv1, ibm-replicaServerId=ldaptiv2- serverid-uid ,ibm-replicaGroup=default, o=ibm,
c=us
objectclass: top
objectclass: ibm-replicationAgreement cn: ldaptiv1
ibm-replicaConsumerId: ldaptiv1-serverid
ibm-replicaUrl: ldap://ldaptiv1:389
ibm-replicaCredentialsDN: cn=simple, cn=replication, cn=IBMpolicies
description: supplier agreement from replication site2 to replication site 1
```

```
dn: cn=ldaptiv2,ibm-replicaServerId=ldaptiv1-serverid,ibm-replicaGroup=default,o=ibm,c=us
objectclass: top
objectclass: ibm-replicationAgreement cn: ldaptiv2
ibm-replicaConsumerId: ldaptiv2-serverid
ibm-replicaUrl: ldap://ldaptiv2:389
ibm-replicaCredentialsDN: cn=simple, cn=replication, cn=IBMpolicies
description: supplier agreement from replication site1 to replication site2
```

14.6 Migration considerations

In this section, we consider different migration scenarios for different software versions.

14.6.1 Tivoli Directory Server v3.2.2 to Tivoli Directory Server v6

The migration considerations for Tivoli Directory Server v3.2.2 to Tivoli Directory Server v6 are:

- With the availability of Tivoli Directory Server 5.1, replication agreements have been rewritten from using replicaObject entries to ibm-replicationContext, ibm-replicaGroup, and ibm-replicaSubentry entries that reside under the customer suffix and not in cn=localhost.
- Use db2ldif to dump the Tivoli Directory Server 3.2.2 database when all replication changes are completed. Check the DB2 change table, it will say “0” changes.
- Install new Tivoli Directory Server v6.
- Load with ldsldif2db or idsbulkload to load the data into Tivoli Directory Server v6.
- Make or add any new replication peers or replicas.

14.6.2 Tivoli Directory Server v4.1 to Tivoli Directory Server v6 (in place)

The migration considerations for Tivoli Directory Server v4.1 to Tivoli Directory Server v6 (in place) are:

- The server updates the cn=Master Server entry in the ibmslapd.conf file.
- Auxiliary class ibm-slapdPendingMigration is added to the cn=Master Server entry to indicate that replica migration should be done during the initial startup of the server.
- The value of the ibm-slapdMigrationInfo attribute indicates what type of server is being migrated. The following are valid values for this attribute:
  - 4.1 REPLICA Read-only replica
  - 4.1 MASTER Read-write master
  - 4.1 PEER Read-write peer replica
- For a read-only replica server, an ibm-replicationContext, ibm-replicaGroup, and ibm-replicaSubentry are created for each suffix that is configured for the server, except for the CN=SCHEMA, CN=LOCALHOST, and CN=PWDPOLICY suffixes.
- For a peer or master server, an ibm-replicationContext, ibm-replicaGroup, and ibm-replicaSubentry are created only if the replicaObject entries exist under the cn=localhost subtree.
- For peer and master servers, all replicaObject entries under the cn=localhost subtree are migrated to ibm-replicationAgreement and ibm-replicationCredentials directory entries.

- For master and peer servers with replicaObject entries, convert currently outstanding replication data and status in the CHANGE and PROGRESS tables to the newly defined Tivoli Directory Server 6.0 REPLCHANGE, REPLSTATUS, and REPLCSTAT tables.
- When the replication is successful, the ibm-slapdPendingMigration auxiliary class is removed from the cn=Master Server entry in the ibmslapd.conf file, and the obsolete CHANGE and PROGRESS tables are deleted from the database.

14.6.3 Tivoli Directory Server v4.1 to Tivoli Directory Server v6 (new servers)

The migration considerations for Tivoli Directory Server v4.1 to Tivoli Directory Server v6 (new servers) are:

- Run **db2ldif** to create a data dump of the 4.1 server. Move this file over to the new Tivoli Directory Server v6.
- Create a migrate directory and copy all the V3.* files and the slapd32.conf file from the 4.1 or run the migbkup bat or ksh file to gather these files. Move this directory to the new Tivoli Directory Server v6.
- Run the **idsimigr** program to create a Tivoli Directory Server v6 instance using the information from the 4.1 migrate directory.
- Run the **idscfgdb** to create the new database and then load the data back with **idsldif2db** or **idsbulkload** tools.
- Create or add new peer or replica agreements.

14.6.4 Tivoli Directory Server v5.1 or v5.2 to Tivoli Directory Server v6

The migration considerations for Tivoli Directory Server v5.1 or v5.2 to Tivoli Directory Server v6 are:

- If you are migrating from IBM Directory Server 5.1 or Tivoli Directory Server 5.2 in place, no migration is required for replication.
- If you are migrating to new servers, you will have to follow the same steps as migrating from Tivoli Directory Server 4.1 using new servers, except you will have an ibmslapd.conf file instead of slapd32.conf file; other than that the process is the same.

14.7 Synchronizing two-way cryptography for server instances

If you want to use replication, use a distributed directory, or import and export LDIF data between server instances, you must cryptographically synchronize the server instances to obtain the best performance.

If you already have a server instance, and you have another server instance that you want to cryptographically synchronize with the first server instance, use the **idsgendirksf** utility to re-create the ibmslapddir.ksf file (the key stash file) from the first server instance. This file is used to replace the second (or more) server instance's original ibmslapddir.ksf file.

You have to perform this procedure (**idsgendirksf**) *before* you do any of the following:

- Start the second server instance.
- Run the **idsbulkload** command from the second server instance.
- Run the **idsldif2db** command from the second server instance.

15

Adding a new LDAP server to an existing enclave

In this chapter, we discuss how to install and add a new Lightweight Directory Access Protocol (LDAP) server into an existing environment including the proper replication configuration and test.

© Copyright IBM Corp. 2007. All rights reserved.

15.1 Installing a new Tivoli Directory Server

Use the native installation procedures for the operating system (OS) that you are going to use. Apply the current DB2 and Tivoli Directory Server fix packs before you configure the new server. You have to build this new server the same way the other LDAP servers were built, using the same paths, instance names, instance user IDs, and group names.

Usually, if you configure a Tivoli Directory Server, it generates an *ibm-slapdServerId* that looks something like this: `4dbdc73a-2476-4039-8808-9655f95d917`. This makes it very hard to troubleshoot when you have a complex enclave with multiple peers, forwarders and/or replicas. Also, when you are building your replication LDIFs for complex agreements, it is better that you use a server ID that better describes the server you are working with. A better way is to develop your own unique server ID.

Before you start the server, you can edit the /export/home/ldapdb2/idsslapd-ldapdb2 /etc/ibmslapd.conf file. Add the *ibm-slapdServerId* parameter with the appropriate server ID and update the lines such as the ibm-slapdDbConnections as shown in Example 15-1.

Important: The server ID specified must match the server ID used in the replication agreement. The recommended format for the server ID is *<shorthostname>-uid*. For example, the short name of a server named *new60ITDSserver_name.tivoli.com* is *new60ITDSserver_name,* therefore the server ID is *new60ITDSserver_name-uid*. Lines represented in bold are to be added. Lines represented in italics are reference points.

Example 15-1 ibmslapd.conf file excerpt

```
ibm-slapdServerBackend: RDBM
ibm-slapdServerId: <shorthostname>-uid
ibm-slapdSizeLimit: 500
. . .
ibm-slapdIdleTimeOut: 300
ibm-slapdSetEnv: DB2CODEPAGE=1208
ibm-slapdSetEnv: LDAP_MAXCARD=YES
. . .
cn: Connection Management
#ibm-slapdAllowAnon: TRUE
ibm-slapdAllowAnon: FALSE
ibm-slapdAllReapingThreshold: 1200
. . .
ibm-slapdDbAlias: ldapdb2b
#ibm-slapdDbConnections: 15
ibm-slapdDbConnections: 30
ibm-slapdDbInstance: ldapdb2
```

15.2 Building new replication agreements

In order to build the proper replication agreements, we have to perform the following steps described in the following sections.

15.2.1 Defining the role of the new Tivoli Directory Server v6

The steps in this section should be performed on each of the new LDAP servers.

1. Log on to the new Tivoli Directory Server v6 and switch to the root user:

   ```
   $ su - root
   ```

   Enter the password.

2. Set the **umask**.

   ```
   # umask 022
   ```

3. Edit the /opt/export/home/ldapdb2/idsslapd-ldapdb2/etc/ibmslapd.conf file and add the contents of the appropriate replication ibmslapd file. Go to the end of the file and add the contents of one of the five files below depending on what type of server you are installing.

   – peer_Ibmslapd.txt

   ```
   dn: cn=Master Server, cn=configuration
   cn: Master Server
   ibm-slapdMasterDN: cn=peermaster
   ibm-slapdMasterPW: <password>
   objectclass: ibm-slapdConfigEntry
   objectclass: ibm-slapdReplication
   objectclass: top
   ```

   – site1_forwarder_Ibmslapd.txt

   ```
   dn: cn=Master Server, cn=configuration
   cn: Master Server
   ibm-slapdMasterDN: cn=peermaster
   ibm-slapdMasterPW: <password>
   ibm-slapdMasterReferral: ldap://<master peer server host name>:389
   objectclass: ibm-slapdConfigEntry
   objectclass: ibm-slapdReplication
   objectclass: top
   ```

   – site1_replica_Ibmslapd.txt

   ```
   dn: cn=Master Server, cn=configuration
   cn: Master Server
   ibm-slapdMasterDN: cn=peermaster
   ibm-slapdMasterPW: <password>
   ibm-slapdMasterReferral: ldap://<forwarder server host name>:389
   objectclass: ibm-slapdConfigEntry
   objectclass: ibm-slapdReplication
   objectclass: top
   ```

   – site2_forwarder_Ibmslapd.txt

   ```
   dn: cn=Master Server, cn=configuration
   cn: Master Server
   ibm-slapdMasterDN: cn=peermaster
   ibm-slapdMasterPW: <password>
   ibm-slapdMasterReferral: ldap://<master peer server host name>:389
   objectclass: ibm-slapdConfigEntry
   ```

```
objectclass: ibm-slapdReplication
objectclass: top
```

– site2_replica_Ibmslapd.txt

```
dn: cn=Master Server, cn=configuration
cn: Master Server
ibm-slapdMasterDN: cn=peermaster
ibm-slapdMasterPW: <password>
ibm-slapdMasterReferral: ldap://<forwarder server host name>:389
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdReplication
objectclass: top
```

15.2.2 Creating the new replication agreement to add the new server

Add the replication agreement and credentials to the server. Because the replication agreement can be replicated, a DN to a credentials object is used. Use of a separate object also makes it easier to support various authentication methods; new object classes can be created rather than trying to make sense of numerous optional attributes.

The examples that are included in this section are:

- Adding a new peer
- Adding a new forwarder
- Adding a new replica

Example 1: Adding a new peer agreement

Before you begin, you have to know what the other existing peer and forwarder LDAP server's host names are. For our example, we use the existing LDAP server's host names, which are peer1, peer2, forwarder1, and forwarder2. The new peer master's host name is peer3 and this is for a Tivoli Access Manager for e-business environment.

With this information at hand, we can build the new agreement to be added to the existing enclave. Create a file called add_peer_input.txt, as shown in Example 15-2.

Example 15-2 add_peer_input.txt

```
#peer3 site1 peer master 3 subentry
#true if master, false if  forwarder
dn: ibm-replicaServerId=peer3-uid, ibm-replicaGroup=default, c=us
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: peer3-uid
ibm-replicationServerIsMaster: true
cn: peer3
description: site1 peer master 3 ibm-replicaSubentry

#peer3 site1 peer master to forwarder1 site1 forwarder agreement
dn: cn=forwarder, ibm-replicaServerId=peer3-uid, ibm-replicaGroup=default, c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: forwarder1
ibm-replicaConsumerId: forwarder1-uid
```

```
ibm-replicaUrl: ldap://forwarder1:389
ibm-replicaCredentialsDN: cn=ibmcred, cn=replication, cn=IBMpolicies
description: peer3 (site1 peer master 3) to forwarder1 (the site1 forwarder)
agreement

#peer3 site1 peer master 3 to forwarder2 site2 forwarder agreement
dn: cn=forwader2,ibm-replicaServerId=peer3-uid, ibm-replicaGroup=default, c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: forwarder2
ibm-replicaConsumerId: forwarder2-uid
ibm-replicaUrl: ldap://forwarder2:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer3 (site1 peer master 3) to forwarder2 (the site2 forwarder)
agreement

#peer1 site1 peer master 1 to peer3 site1 peer master 3 agreement
dn: cn=peer3,ibm-replicaServerId=peer1-uid, ibm-replicaGroup=default, c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer3
ibm-replicaConsumerId: peer3-uid
ibm-replicaUrl: ldap://peer3:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer1 (site1 peer master 1) to peer3 (site1 peer master 3) agreement

#peer2 site2 peer master 2 to peer3 site1 peer master 1 agreement
dn: cn=peer3,ibm-replicaServerId=peer2-uid, ibm-replicaGroup=default, c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer3
ibm-replicaConsumerId: peer3-uid
ibm-replicaUrl: ldap://peer3:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer2  (site2 peer master 2) to peer3 (site1 peer master 3) agreement

#peer3 site1 peer master 3 subentry
#true if master, false if forwarder
dn: ibm-replicaServerId=peer3-uid,ibm-replicaGroup=default,cn=ibmpolicies
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: peer3-uid
ibm-replicationServerIsMaster: true
cn: peer3
description: site1 peer master 3 ibm-replicaSubentry

#peer3 site1 peer master to forwarder1 site1 forwarder agreement
dn: cn=forwarder1, ibm-replicaServerId=peer3-uid, ibm-replicaGroup=default,
cn=ibmpolicies
objectclass: top
objectclass: ibm-replicationAgreement
cn: forwarder1
ibm-replicaConsumerId: forwarder1-uid
ibm-replicaUrl: ldap://forwarder1:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
```

```
description: peer3 (site1 peer master 3) to forwarder1 (the site1 forwarder)
agreement

#peer3 site1 peer master 3 to forwarder2 site2 forwarder agreement
dn: cn=forwader2,ibm-replicaServerId=peer3-uid, ibm-replicaGroup=default,
cn=ibmpolicies
objectclass: top
objectclass: ibm-replicationAgreement
cn: forwarder2
ibm-replicaConsumerId: forwarder2-uid
ibm-replicaUrl: ldap://forwarder2:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer3 (site1 peer master 3) to forwarder2 (the site2 forwarder)
agreement

#peer1 site1 peer master 1 to peer3 site1 peer master 3 agreement
dn: cn=peer3,ibm-replicaServerId=peer1-uid, ibm-replicaGroup=default,
cn=ibmpolicies
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer3
ibm-replicaConsumerId: peer3-uid
ibm-replicaUrl: ldap://peer3:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer1 (site1 peer master 1) to peer3 (site1 peer master 3) agreement

#peer2 site2 peer master 2 to peer3 site1 peer master 1 agreement
dn: cn=peer3,ibm-replicaServerId=peer2-uid, ibm-replicaGroup=default,
cn=ibmpolicies
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer3
ibm-replicaConsumerId: peer3-uid
ibm-replicaUrl: ldap://peer3:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer2  (site2 peer master 2) to peer3 (site1 peer master 3) agreement

#peer3 site1 peer master 3 subentry
#true if master, false if forwarder
dn: ibm-replicaServerId=peer3-uid,ibm-replicaGroup=default,secauthority=default
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: peer3-uid
ibm-replicationServerIsMaster: true
cn: peer3
description: site1 peer master 3 ibm-replicaSubentry

#peer3 site1 peer master to forwarder1 site1 forwarder agreement
dn: cn=forwarder1, ibm-replicaServerId=peer3-uid, ibm-replicaGroup=default,
secAuthority=default
objectclass: top
objectclass: ibm-replicationAgreement
cn: forwarder1
ibm-replicaConsumerId: forwarder1-uid
ibm-replicaUrl: ldap://forwarder1:389
```

```
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer3 (site1 peer master 3) to forwarder1 (the site1 forwarder)
agreement

#peer3 site1 peer master 3 to forwarder2 site2 forwarder agreement
dn: cn=forwader2,ibm-replicaServerId=peer3-uid, ibm-replicaGroup=default,
secAuthority=default
objectclass: top
objectclass: ibm-replicationAgreement
cn: forwarder2
ibm-replicaConsumerId: forwarder2-uid
ibm-replicaUrl: ldap://forwarder2:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer3 (site1 peer master 3) to forwarder2 (the site2 forwarder)
agreement

#peer1 site1 peer master 1 to peer3 site1 peer master 3 agreement
dn: cn=peer3,ibm-replicaServerId=peer1-uid, ibm-replicaGroup=default,
secAuthority=default
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer3
ibm-replicaConsumerId: peer3-uid
ibm-replicaUrl: ldap://peer3:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer1 (site1 peer master 1) to peer3 (site1 peer master 3) agreement

#peer2 site2 peer master 2 to peer3 site1 peer master 1 agreement
dn: cn=peer3,ibm-replicaServerId=peer2-uid, ibm-replicaGroup=default,
secAuthority=default
objectclass: top
objectclass: ibm-replicationAgreement
cn: peer3
ibm-replicaConsumerId: peer3-uid
ibm-replicaUrl: ldap://peer3:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer2  (site2 peer master 2) to peer3 (site1 peer master 3) agreement
```

Example 2: Adding a new forwarder agreement

Before you begin, you have to know what the other existing peer and forwarder LDAP server's host names are. For our example, we use the existing LDAP server's host names, which are peer1, peer2, forwarder1, and forwarder2. The new forwarder host name is forwarder3 (see "Example 3: Adding a new replica agreement" on page 107, for adding a new replica to a forwarder) and this is for a Tivoli Access Manager for e-business environment.

With this information at hand, we can build the new agreement to be added to the existing enclave. Create a file called add_forwarder_input.txt, as shown in Example 15-3.

Example 15-3 add_forwarder_input.txt

```
#forwarder3 site1 forwarder subentry
#true if master, false if  forwarder
dn: ibm-replicaServerId=forwarder3-uid, ibm-replicaGroup=default, c=us
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: forwarder3-uid
ibm-replicationServerIsMaster: false
cn: forwarder3
description: site1 forwarder ibm-replicaSubentry

#peer1 site1 peer master to forwarder3 site1 forwarder 3 agreement
dn: cn=forwarder3, ibm-replicaServerId=peer1-uid, ibm-replicaGroup=default, c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: forwarder3
ibm-replicaConsumerId: forwarder3-uid
ibm-replicaUrl: ldap://forwarder3:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer1 (site1 peer master 1) to forwarder3 (the site1 forwarder 3)
agreement

#peer2 site2 peer master 2 to forwarder3 site1 forwarder 3 agreement
dn: cn=forwarder3, ibm-replicaServerId=peer2-uid, ibm-replicaGroup=default, c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: forwarder3
ibm-replicaConsumerId: forwarder3-uid
ibm-replicaUrl: ldap://forwarder3:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer2 (site2 peer master 2) to forwarder3 (the site1 forwarder 3)
agreement

#forwarder3 site1 forwarder subentry
#true if master, false if forwarder
dn: ibm-replicaServerId=forwarder3-uid,ibm-replicaGroup=default,cn=ibmpolicies
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: forwarder3-uid
ibm-replicationServerIsMaster: false
cn: forwarder3
description: site1 forwarder ibm-replicaSubentry

#peer1 site1 peer master to forwarder3 site1 forwarder 3 agreement
dn: cn=forwarder3, ibm-replicaServerId=peer1-uid, ibm-replicaGroup=default,
cn=ibmpolicies
objectclass: top
objectclass: ibm-replicationAgreement
cn: forwarder3
ibm-replicaConsumerId: forwarder3-uid
ibm-replicaUrl: ldap://forwarder3:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer1 (site1 peer master 1) to forwarder3 (the site1 forwarder 3)
agreement
```

```
#peer2 site2 peer master 2 to forwarder3 site1 forwarder 3 agreement
dn: cn=forwarder3, ibm-replicaServerId=peer2-uid, ibm-replicaGroup=default,
cn=ibmpolicies
objectclass: top
objectclass: ibm-replicationAgreement
cn: forwarder3
ibm-replicaConsumerId: forwarder3-uid
ibm-replicaUrl: ldap://forwarder3:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer2 (site2 peer master 2) to forwarder3 (the site1 forwarder 3)
agreement

#forwarder3 site1 forwarder subentry
#true if master, false if forwarder
dn:
ibm-replicaServerId=forwarder3-uid,ibm-replicaGroup=default,secauthority=default
objectclass: top
objectclass: ibm-replicaSubentry
ibm-replicaServerId: forwarder3-uid
ibm-replicationServerIsMaster: false
cn: forwarder3
description: site1 forwarder ibm-replicaSubentry

#peer1 site1 peer master to forwarder3 site1 forwarder 3 agreement
dn: cn=forwarder3, ibm-replicaServerId=peer1-uid, ibm-replicaGroup=default,
secAuthority=default
objectclass: top
objectclass: ibm-replicationAgreement
cn: forwarder3
ibm-replicaConsumerId: forwarder3-uid
ibm-replicaUrl: ldap://forwarder3:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer1 (site1 peer master 1) to forwarder3 (the site1 forwarder 3)
agreement

#peer2 site2 peer master 2 to forwarder3 site1 forwarder 3 agreement
dn: cn=forwarder3, ibm-replicaServerId=peer2-uid, ibm-replicaGroup=default,
secAuthority=default
objectclass: top
objectclass: ibm-replicationAgreement
cn: forwarder3
ibm-replicaConsumerId: forwarder3-uid
ibm-replicaUrl: ldap://forwarder3:389
ibm-replicaCredentialsDN: cn= ibmcred, cn=replication, cn=IBMpolicies
description: peer2 (site2 peer master 2) to forwarder3 (the site1 forwarder 3)
agreement
```

Example 3: Adding a new replica agreement

Before you begin, you have to know what the other existing forwarder LDAP server's host name that you are adding this replica. For our example, we use the existing LDAP forwarder server, which is forwarder1 with the other replicas configured. The new replica host name is replica3 and this is for a Tivoli Access Manager for e-business environment.

With this information at hand, we can build the new agreement to be added to the existing enclave. Create a file called add_replica_input.txt as shown in Example 15-4.

Example 15-4 add_replica_input.txt

```
#forwarder1 site1 forwarder to replica3 site1 replica agreement
dn: cn=replica3, ibm-replicaServerId=forwarder1-uid, ibm-replicaGroup=default,
c=us
objectclass: top
objectclass: ibm-replicationAgreement
cn: replica3
ibm-replicaConsumerId: replica3-uid
ibm-replicaUrl: ldap://replica3:389
ibm-replicaCredentialsDN: cn=ibmcred, cn=replication, cn=IBMpolicies
description: forwarder1 (the site1 forwarder) to replica3 (the site1 replica 3)
agreement

#forwarder1 site1 forwarder to replica3 site1 replica agreement
dn: cn=replica3, ibm-replicaServerId=forwarder1-uid, ibm-replicaGroup=default,
cn=ibmpolicies
objectclass: top
objectclass: ibm-replicationAgreement
cn: replica3
ibm-replicaConsumerId: replica3-uid
ibm-replicaUrl: ldap://replica3:389
ibm-replicaCredentialsDN: cn=ibmcred, cn=replication, cn=IBMpolicies
description: forwarder1 (the site1 forwarder) to replica3 (the site1 replica 3)
agreement

#forwarder1 site1 forwarder to replica3 site1 replica agreement
dn: cn=replica3, ibm-replicaServerId=forwarder1-uid, ibm-replicaGroup=default,
secAuthority=default
objectclass: top
objectclass: ibm-replicationAgreement
cn: replica3
ibm-replicaConsumerId: replica3-uid
ibm-replicaUrl: ldap://replica3:389
ibm-replicaCredentialsDN: cn=ibmcred, cn=replication, cn=IBMpolicies
description: forwarder1 (the site1 forwarder) to replica3 (the site1 replica 3)
agreement
```

15.2.3 Loading the new agreement

In order to load the new agreement, perform the following steps:

1. Make sure that the existing LDAP enclave is up and running with no replication problems.
2. Log on to one of the peer master servers.
3. Enter the following command to load the new agreement into the existing enclave. (XXXX = name of the replication agreement that you will be using).

   ```
   ldapmodify -p 389 -D cn=root -w <password> -c -f /opt/tmp/XXXX_input.txt
   ```

Note: The XXXX_input.txt files are the sample files above that have to be customized for each Tivoli Access Manager for e-business LDAP enclave.

4. Check to make sure that the new agreements have been replicated throughout the enclave. You can use the **ldapsearch** command to look for the changes in all the existing servers.

15.2.4 Backing up data from a Tivoli Directory Server v6 peer master server

To back up data from a Tivoli Directory Server v6 peer master server:

1. Log on to one of the Tivoli Directory Server v6 peer master servers and switch to the root user:

   ```
   $ su - root
   ```

   Enter the password.

2. Set the **umask**.

   ```
   # umask 022
   ```

3. Switch to the ldapdb2 user.

   ```
   # su - ldapdb2
   ```

4. Run the **tune_runstats.sh** script (covered in 7.2, “How to use tune_runstats.sh” on page 57) to update the statistics.

5. Stop the Tivoli Directory Server v6.

   Note: The “-I” is a capital letter “i”.

   ```
   # idsslapd -I ldapdb2 -k
   ```

   The output should look similar to the following:

   ```
   GLPSRV121I Stopped directory server instance: 'ldapdb2'
   ```

6. Create a directory to store the LDAP data.

   ```
   # mkdir /opt/tmp/dbback
   ```

7. Change the ownership and permissions on /opt/tmp/dbback. The ldapdb2 and dbsysadm group require full access to the dbback directory.

   ```
   # chmod 770 /opt/tmp/dbback
   # chown ldapdb2:dbsysadm /opt/tmp/dbback
   ```

8. Back up the existing Tivoli Directory Server v6 directory.

   Note: The “-I” is a capital letter “i”.

   ```
   # idsdbback -I ldapdb2 -k /opt/tmp/dbback
   ```

The output should look similar to the following:

```
You have chosen to perform the following actions:

GLPDBB029I The database and configuration files for directory server instance
'ldapdb2' will be backed up to '/opt/tmp/dbback'.

Do you want to....
(1) Continue with the above actions, or
(2) Exit without making any changes:
```

9. Enter 1 and press Enter.

   The output looks similar to the output shown in Example 15-5.

Example 15-5 Output for data backup

```
GLPDBB008I Backing up directory server instance 'ldapdb2'.
GLPDBB015I Backing up the configuration file for the directory server instance
'ldapdb2'.
GLPDBB016I Backed up the configuration file for the directory server instance
'ldapdb2'.
GLPDBB018I Backing up the key stash files for the directory server instance
'ldapdb2'.
GLPDBB019I Backed up the key stash files for the directory server instance
'ldapdb2'.
GLPDBB021I Backing up the schema files for the directory server instance
'ldapdb2'.
GLPDBB022I Backed up the schema files for the directory server instance
'ldapdb2'.
GLPCTL008I Starting database manager for database instance: 'ldapdb2'.
GLPCTL009I Started database manager for database instance: 'ldapdb2'.
GLPCTL098I Backing up database ldapdb2.
GLPCTL099I Backed up database ldapdb2.
GLPDBB026I Saving backup information to a file.
GLPDBB027I Saved backup information to a file.
GLPDBB009I Backed up directory server instance 'ldapdb2'.
```

10. Tar the data from the Tivoli Directory Server v6 master server.

```
# cd /opt/tmp
# tar cvf /opt/tmp/ids60backup.tar dbback
```

11. Compress the tar file. The output file will be named /opt/tmp/ids60backup.tar.gz.

```
# gzip /opt/tmp/ids60backup.tar
```

15.2.5 Restoring data to the replicas, peer masters, and forwarders

Important: Do *not* start any of the Tivoli Directory Server 6.0 LDAP servers until the following steps are completed on all servers.

To restore data to the replicas, peer masters, and forwarders:

1. Log on to the new Tivoli Directory Server v6 peer, replica, or forwarder server and switch to the root user:

```
$ su - root
```

Enter the password.

2. Set the **umask**.

   ```
   # umask 022
   ```

3. Transfer/FTP the ids60backup.tar.gz file from the Tivoli Directory Server v6 peer master server /opt/tmp directory to the /opt/tmp directory on the new Tivoli Directory Server v6 peer, replica, or forwarder server.
4. Extract the Tivoli Directory Server v6 master server data from the compressed file.

   ```
   # cd /opt/tmp
   # gunzip /opt/tmp/ids60backup.tar.gz
   ```

5. Untar the /opt/tmp/ids60backup.tar file into the /opt/tmp directory.

   ```
   # tar xvf /opt/tmp/ids60backup.tar
   ```

6. Copy /opt/tmp/dbback/ibmslapddir.ksf /opt/export/home/ldapdb2/idsslapd-ldapdb2/etc.

   ```
   cp /opt/tmp/seed/ibmslapddir.ksf /export/home/ldapdb2/idsslapd-ldapdb2/etc
   ```

7. Restore the data into the LDAP database. Using -r prevents the ibmslapd.conf file from being overwritten by the ibmslapd.conf from the Tivoli Directory Server v6 master server.

   Note: The “-I” is a capital letter “i”.

   ```
   # idsdbrestore -I ldapdb2 -k /opt/tmp/dbback -r
   ```

   The output is similar to the following:

   ```
   You have chosen to perform the following actions:

   GLPDBR026I The database and configuration files for directory server instance
   'ldapdb2' will be restored from files in directory '/opt/tmp/dbback'. Note: The
   data in the currently configured database will be overwritten and will be lost.
   The schema files and directory key stash file currently configured will be
   overwritten. Unless the -r option was specified the configuration file and
   configuration key stash file will also be overwritten.

   Do you want to....
   (1) Continue with the above actions, or
   (2) Exit without making any changes:
   ```

8. Enter `1` and press Enter.

   The output is similar to the one shown in Example 15-6.

   Example 15-6 Output for data restore

   ```
   GLPDBR002I Restoring directory server instance 'ldapdb2'.
   GLPDBR005I Loading backup information from a file.
   GLPDBR006I Loaded backup information from a file.
   GLPCTL008I Starting database manager for database instance: 'ldapdb2'.
   GLPCTL009I Started database manager for database instance: 'ldapdb2'.
   GLPCTL101I Restoring backup database ldapdb2 to configured database ldapdb2.
   GLPCTL102I Restored backup database ldapdb2 to configured database ldapdb2.
   GLPDBR015I Restoring the key stash files for the directory server instance
   'ldapdb2'.
   GLPDBR016I Restored the key stash files for the directory server instance
   'ldapdb2'.
   ```

```
GLPDBR018I Restoring the schema files for the directory server instance
'ldapdb2'.
GLPDBR019I Restored the schema files for the directory server instance
'ldapdb2'.
GLPDBR003I Restored directory server instance 'ldapdb2'.
```

15.2.6 Starting all new LDAP servers and verifying replication queues

Important: Start the LDAP servers *only* after **idsdbrestore** has been run on all *new* peer master, forwarder, and replica servers.

Repeat this section for each new peer master, forwarder, and replica in the enclave. Start the servers in the following order:

1. New peer master servers
2. New forwarders
3. New replicas

Now perform the following steps:

1. Log on to the new Tivoli Directory Server v6 and switch to the root user:

   ```
   $ su - root
   ```

   Enter the password.

2. Set the **umask**.

   ```
   # umask 022
   ```

3. Start the Tivoli Directory Server v6.

   Note: The “-I” is a capital letter “i”.

   ```
   # idsslapd -I ldapdb2
   ```

   The output is similar to the following:

   ```
   GLPSRV041I Server starting.
   GLPCOM024I The extended Operation plugin is successfully loaded from
   libevent.so.
   GLPCOM024I The extended Operation plugin is successfully loaded from
   libtranext.so.
   . . .
   GLPSRV009I IBM Tivoli Directory (SSL), 6.0      Server started.
   ```

4. Verify that the LDAP server is running.

   ```
   # ldapsearch -p 389 -D cn=root -w ldap4u -s base objectclass=*
   ```

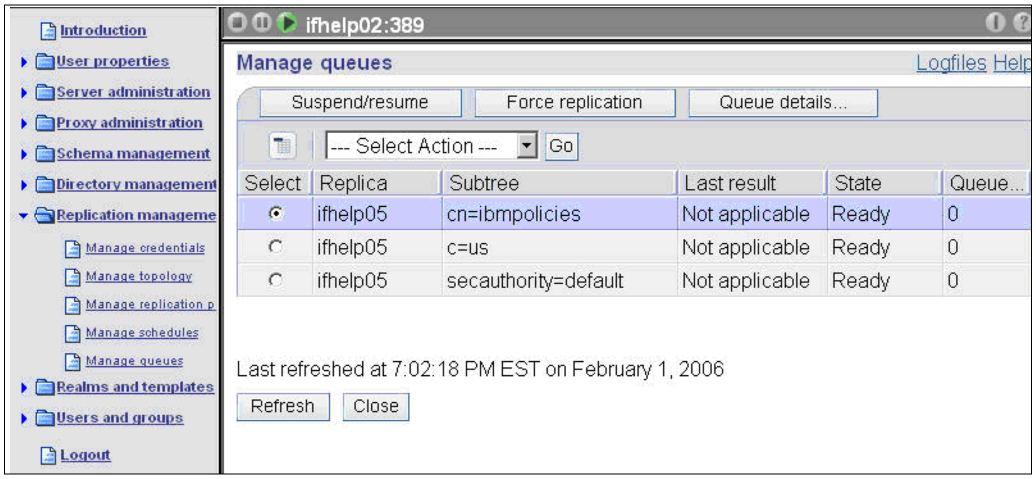
   The beginning and the end of the output should look like the following:

   ```
   namingcontexts=CN=SCHEMA
   . . .
   ibm-slapdisconfigurationmode=FALSE
   ```

5. **Important:** Verify that the last line, *ibm-slapdisconfigurationmode*, has the value of FALSE. If it is TRUE, then the server is in configuration mode because there is a problem. Fix any problems and restart the server before going to the next step.

6. Log on to the LDAP server using the Web administration tool console.

7. Click **Replication Management** → **Manage queues**, as shown in Figure 15-1.



Manage queues

Select	Replica	Subtree	Last result	State	Queue...
●	ifhelp05	cn=ibmpolicies	Not applicable	Ready	0
○	ifhelp05	c=us	Not applicable	Ready	0
○	ifhelp05	secauthority=default	Not applicable	Ready	0

Last refreshed at 7:02:18 PM EST on February 1, 2006

Figure 15-1 Manage replication queues

8. Verify that the state is *Ready* for each server and suffix combination. If you are logged into a peer server, there should be queues for each of the other peer servers and the forwarders. If you are logged into a forwarder, there should be queues for each of the replicas under that forwarder. If you are logged into a replica, there should be no queues.

15.3 Testing replication

Replication is tested by creating test entries under c=US. Each peer master server should perform at least one write action to test replication. The new entries are propagated to the forwarder and then to the replicas. The entries are deleted when the test is complete (these script files are located in the test folder).

1. Log on to the original Tivoli Directory Server v6 master server and switch to the root user:

```
$ su - root
```

   Enter the password.

2. Set the **umask**.

```
# umask 022
```

3. Add a test organization to c=US using a peer master server. The ldif entry for this organization has been provided in /opt/tmp/myTestOrg1.ldif.

```
# cd /opt/tmp/tools/replication/test
# ldapadd -D cn=root -w <password> -h <peer_master1> -p 389 -f myTestOrg1.ldif
```

4. Search for the test organization on a replica server.

```
# ldapsearch -D cn=root -w <password> -h <replica> -p 389 -b c=us o=myTestOrg1
```

   The output is similar to the following:

```
o=myTestOrg1, c=us
o=myTestOrg1
objectclass=top
objectclass=organization
```

Note: Repeat the following steps (add/search) using a different number for each peer master.

5. Add a test user to o=myTestOrg1, c=US using a different peer master server. The ldif entry for this organization has been provided in /opt/tmp/tools/replication/myTestPerson<N>.ldif, where <N> is equal to 1, 2, 3, 4, or 5. Use a different file for each master server.

```
# cd /opt/tmp/tools/replication/test
# ldapadd -D cn=root -w <password> -h <peer_masterN> -p 389 -f myTestPerson
<N>.ldif
```

6. Search a replica server for the test user created in the previous step. If an entry is not returned, wait a few minutes and try again. If there is still no entry, then there is a problem with the replication. Fix any problems before continuing.

Note: myTestPerson<N> corresponds to the file used to create the entry, where <N> is equal to 1, 2, 3, 4, or 5.

```
# ldapsearch -D cn=root -w <password> -h <replica> -p 389 -b c=us
cn=myTestPerson<N>
```

The output is similar to the following:

```
cn=myTestPerson1, o=myTestOrg1, c=us
cn=myTestPerson1
sn=myPersonSn1
objectclass=top
objectclass=person
```

7. Delete the test entries that were created.

```
# cd /opt/tmp/tools/replication/test
# ldapdelete -D cn=root -w <password> -h <peer_master> -p 389 -c -f
myTestDelete.ldif
```

If all five of the *myTestPersons* were added, the output will be blank. For each *myTestPerson* that was *not* added, *No such object* will be reported. Assuming three peer masters are used and one user was added to each peer master, the output is similar to the following:

```
ldap_delete_s: No such object

ldap_delete_s: No such object
```

8. Verify that the test entries have been removed.

```
# ldapsearch -D cn=root -w <password> -h <replica> -p 389 -b "o=myTestOrg1,
c=us"  objectclass=*
```

There should be no output.

A

Special operating system tuning for Tivoli Directory Server

In this appendix, we look into specific tuning for the HP-UX, Sun Solaris, and IBM AIX operating systems (OS) as well as some other OS-related environment variables.

Sun Solaris and HP-UX operating system tuning

DB2 has a tool called *db2osconf* that can make recommendations for kernel parameter values based on the size of a system. The recommended values are high enough for a given system that they can accommodate most reasonable workloads. This command is currently available only for DB2 on HP-UX on 64-bit instances and the Sun Solaris operating environment.

- On DB2 for HP-UX, no authorization is required. To make the changes recommended by the db2osconf utility, you must have root access.
- On DB2 for the Sun Solaris operating environment, you must have root access or be a member of the sys group.

Determining which system settings are required for DB2 and LDAP

To determine the system settings required for DB2 and LDAP:

1. Enter the following command:

   ```
   # /opt/IBM/db2/V8.1/bin/db2osconf -x
   ```

   The results depend on the size of the server and will vary from one machine to another. The output is similar to the following.

   ```
   set msgsys:msginfo_msgmax = 65535
   set msgsys:msginfo_msgmnb = 65535
   set msgsys:msginfo_msgmni = 1792
   set msgsys:msginfo_msgtql = 1792
   set semsys:seminfo_semmni = 2048
   set semsys:seminfo_semmns = 4300
   set semsys:seminfo_semmnu = 2048
   set semsys:seminfo_semume = 240
   set shmsys:shminfo_shmmax = 933521817
   set shmsys:shminfo_shmmni = 2048
   set shmsys:shminfo_shmseg = 240

   Total kernel space for IPC:
   0.28MB (shm) + 1.41MB (sem) + 1.31MB (msg) == 3.01MB (total)
   Do you want to accept the above recommended kernel parameters? (y/n)?
   ```

2. Enter `y` and press Enter.

   The results depend on the size of the server and varies from one machine to another. The output is similar to the following.

   ```
   msgmax =             65535 (old:             2048)
   msgmnb =             65535 (old:             4096)
   msgtql =              1792 (old:               40)
   msgmni =              1792 (old:               50)
   semmni =              2048 (old:               10)
   semmns =              4300 (old:               60)
   semmnu =              2048 (old:               30)
   semume =               240 (old:               10)
   shmmax =         933521817 (old:          1048576)
   shmmni =              2048 (old:              100)
   shmseg =               240 (old:                6)

   Total kernel space for IPC:
   0.28MB (shm) + 1.41MB (sem) + 1.31MB (msg) == 3.01MB (total)
   Do you want to change now to your new kernel parameters? (y/n)?
   ```

3. Enter `y` and press Enter.

   The output is similar to the following:

```
Please backup /etc/system and then copy /tmp/DB2system to /etc/system, then
reboot before further installation.
Backup /etc/system.

# cp /etc/system /etc/system.old

Replace /etc/system with the /tmp/DB2system file created by db2osconf.

# cp /tmp/DB2system /etc/system
```

4. Reboot the server to activate the new system settings.

```
# sync
# sync
# reboot
```

IBM AIX operating system tuning

This appendix discusses the following performance tuning tasks for the AIX operating system:

- Enabling large files
- Setting MALLOCTYPE
- Setting other environment variables
- Viewing ibmslapd environment variables

Enabling large files

The underlying AIX operating system files that hold the contents of a large directory can grow beyond the default size limits imposed by the AIX operating system. If the size limits are reached, the directory ceases to function correctly. The following steps make it possible for files to grow beyond default limits on an AIX operating system:

- When you create the file systems that are expected to hold the directory's underlying files, you should create them as *Enhanced Journaled File Systems* or as *Journaled File Systems with Large File Enabled*. The file system containing the DB2 instance's home directory, and, if bulkload is used, the file system containing the bulkload temporary directory, are file systems that can be created this way.

  Note: The default path is <instance_home>/tmp.

- Set the soft file size limit for the root, Lightweight Directory Access Protocol (LDAP), and the DB2 instance owner users to -1. A soft file size limit of -1 for a user specifies the maximum file size for that user as unlimited. The soft file size limit can be changed using the **`smitty chuser`** command. Each user must log off and log back in for the new soft file size limit to take effect. You must also restart DB2.

Setting MALLOCTYPE

Malloc buckets provides an optional buckets-based extension of the default allocator. It is intended to improve malloc performance for applications that issue large numbers of small allocation requests. When malloc buckets is enabled, allocation requests that fall within a predefined range of block sizes are processed by malloc buckets. All other requests are processed in the usual manner by the default allocator.

A bucket consists of a block of memory that is subdivided into a predetermined number of smaller blocks of uniform size, each of which is a unit of memory that can be allocated. Each bucket is identified using a bucket number. The first bucket is bucket zero, the second bucket is bucket one, and the third bucket is bucket two, and so on. The first bucket is the smallest and each bucket after that is larger in size than the preceding bucket, using a formula described later in this section. A maximum of 128 buckets is available per heap.

Set the MALLOCTYPE environment variable.

On all AIX 5.x versions, set MALLOCTYPE as follows:

```
export MALLOCTYPE=buckets
```

Note: If you want to use MALLOCTYPE buckets, you must use ML03 (contains the fix for APAR IY50668) or later. You can get this from IBM Support at the following Web site:

http://www.ibm.com/support/us/

If you are using MALLOCTYPE buckets, you must set *ulimits* for the LDAP instance to the following:

```
# ulimit -m unlimited
# ulimit -d unlimited
```

You can find more information about MALLOCTYPE in the AIX product documentation.

Setting other environment variables

You might experience better performance by tuning the following environment variables as shown in the following sections. Check the AIX documentation to see if these settings might be right for your installation.

AIXTHREAD_SCOPE

On AIX, when using multi-threaded applications, especially when running on machines with multiple CPUs, we strongly recommend setting AIXTHREADSCOPE=S in the environment before starting the application, for better performance and more solid scheduling.

Setting AIXTHREAD_SCOPE=S means that user threads created with default attributes are placed into system-wide contention scope. If a user thread is created with system-wide contention scope, it is bound to a kernel thread and it is scheduled by the kernel. The underlying kernel thread is not shared with any other user thread. S signifies system-based contention scope (1:1).

To set AIXTHREAD_SCOPE, use the following command:

```
export AIXTHREAD_SCOPE=S
```

Permanent change is made by adding the AIXTHREAD_SCOPE=S command to the /etc/environment file.

NODISCLAIM

NODISCLAIM controls how calls to free() are being handled. When PSALLOC is set to early, all free() calls result in a disclaim() system call. When NODISCLAIM is set to true, this does not occur.

To set NODISCLAIM, use the following command:

```
export NODISCLAIM=TRUE
```

Permanent change is made by adding the AIXTHREAD_SCOPE=S command to the /etc/environment file.

Setting MINPERM and MAXPERM settings

On AIX servers, there is a default value for MINPERM and MAXPERM that is set as part of a regular build. These values are not optimal for systems that host databases or LDAP directories. The reason is that the default setting is to help cache file systems; DB2 also caches data for performance. Therefore, using the default MINPERM and MAXPERM means that we are double caching quite unnecessarily.

Recommended values

As a rule of thumb, the following values can be used for database servers that run on AIX.

- MINPERM - 10%
- MAXPERM - 20%
- MAXCLIENT - 20%

Note: MAXPERM and MAXCLIENT have to be *identical* values.

Values for MINPERM and MAXPERM parameters

The operating system takes advantage of the varying requirements for real memory by leaving in memory pages of files that have been read or written. If the file pages are requested again before their page frames are reassigned, this technique saves an I/O operation. These file pages might be from local or remote (for example, NFS) file systems.

The ratio of page frames used for files versus those used for computational (working or program text) segments is loosely controlled by the MINPERM and MAXPERM values:

- If percentage of RAM occupied by file pages rises above MAXPERM, page-replacement steals only file pages.
- If percentage of RAM occupied by file pages falls below MINPERM, page-replacement steals both file and computational pages.
- If percentage of RAM occupied by file pages is between MINPERM and MAXPERM, page-replacement steals only file pages unless the number of file repages is higher than the number of computational repages.

In a particular workload, it might be worthwhile to emphasize the avoidance of file I/O. In another workload, keeping computational segment pages in memory might be more important. To understand what the ratio is in the untuned state, use the **vmstat** command with the -v option. For details, see Example A-1.

Example: A-1 vmstat -v output

```
# vmstat -v
              1048576 memory pages
              1002054 lruable pages
               478136 free pages
                    1 memory pools
                95342 pinned pages
                 80.1 maxpin percentage
                 20.0 minperm percentage
                 80.0 maxperm percentage
```

```
         36.1 numperm percentage
       362570 file pages
          0.0 compressed percentage
            0 compressed pages
         35.0 numclient percentage
         80.0 maxclient percentage
       350782 client pages
            0 remote pageouts scheduled
           80 pending disk I/Os blocked with no pbuf
            0 paging space I/Os blocked with no psbuf
         3312 filesystem I/Os blocked with no fsbuf
            0 client filesystem I/Os blocked with no fsbuf
       474178 external pager filesystem I/Os blocked with no fsbuf
```

The numperm value gives the number of file pages in memory, 362570. This is 36.1% of real memory.

If you notice that the system is paging out to paging space, it can be that the file repaging rate is higher than the computational repaging rate because the number of file pages in memory is below the MAXPERM value. Therefore, in this case we can prevent computational pages from being paged out by lowering the MAXPERM value to something lower than the numperm value. Because the numperm value is approximately 36%, we can lower the MAXPERM value down to 30%. Therefore, the page replacement algorithm only steals file pages. If the lru_file_repage parameter is set to zero, only file pages are stolen if the number of file pages in memory is greater than the value of the MINPERM parameter.

Setting MINFREE and MAXFREE

There is a simple algorithm to calculate values for MINFREE and MAXFREE.

- Find the number of CPUs on the server (#cpus)

  minfree = 120 * #cpus

- Find maxpgahead by using the following command:

  ```
  ioo -a | grep maxpgahead
  ```

  If maxpgahead <= 8 then maxfree = 128 * #cpus

  If maxpgahead > 8 maxfree = new minfree + (#cpus * maxpgahead)

Values for MINFREE and MAXFREE parameters

The purpose of the free list is to keep track of real-memory page frames released by terminating processes and to supply page frames to requestors immediately, without forcing them to wait for page steals and the accompanying I/O to complete. The MINFREE limit specifies the free-list size, below which, page stealing to replenish the free list is to be started. The MAXFREE parameter is the size above which stealing ends. In the case of enabling strict file cache limits, such as the strict_maxperm or strict_maxclient parameters, the MINFREE value is used to start page stealing. When the number of persistent pages is equal to or less than the difference between the values of the MAXPERM and MINFREE parameters, with the strict_maxperm parameter enabled, or when the number of client pages is equal to or less than the difference between the values of the maxclient and MINFREE parameters, with the strict_maxclient parameter enabled, page stealing starts.

The objectives in tuning these limits are to ensure that:

- Any activity that has critical response-time objectives can always get the page frames it needs from the free list.
- The system does not experience unnecessarily high levels of I/O because of premature stealing of pages to expand the free list.

The default values of the MINFREE and MAXFREE parameters depend on the memory size of the machine. The difference between the MINFREE and MAXFREE parameters should always be equal to or greater than the value of the maxpgahead parameter, if you are using Journaled File System (JFS). For Enhanced JFS, the difference between the MAXFREE and MINFREE parameters should always be equal to or greater than the value of the j2_maxPageReadAhead parameter. If you are using both JFS and Enhanced JFS, you should set the value of the MINFREE parameter to a number that is greater than or equal to the larger pageahead value of the two file systems.

The MINFREE and MAXFREE parameter values are different if there is more than one memory pool. Memory pools were introduced in AIX 4.3.3 for MP systems with large amounts of RAM. Each memory pool has its own MINFREE and MAXFREE values, but the MINFREE and MAXFREE values shown by the **vmo** command is the sum of the MINFREE and MAXFREE values for all memory pools.

A less precise but more comprehensive tool for investigating an appropriate size for MINFREE is the **vmstat** command. Example A-2 shows a portion of **vmstat** command output on a system where the MINFREE value is being reached.

Example: A-2 vmstat 1 output

```
# vmstat 1
kthr     memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs   us sy id wa
 2  0 70668   414   0   0   0   0    0   0 178 7364  257   35 14  0 51
 1  0 70669   755   0   0   0   0    0   0 196 19119 272  40 20  0 41
 1  0 70704   707   0   0   0   0    0   0 190 8506  272   37  8  0 55
 1  0 70670   725   0   0   0   0    0   0 205 8821  313   41 10  0 49
 6  4 73362   123   0   5  36 313 1646   0 361 16256 863  47 53  0  0
 5  3 73547   126   0   6  26 152  614   0 324 18243 1248 39 61  0  0
 4  4 73591   124   0   3  11  90  372   0 307 19741 1287 39 61  0  0
 6  4 73540   127   0   4  30 122  358   0 340 20097 970  44 56  0  0
 8  3 73825   116   0  18  22 220  781   0 324 16012 934  51 49  0  0
 8  4 74309    26   0  45  62 291 1079   0 352 14674 972  44 56  0  0
 2  9 75322     0   0  41  87 283  943   0 403 16950 1071 44 56  0  0
 5  7 75020    74   0  23 119 410 1611   0 353 15908 854  49 51  0  0
```

In the output shown in Example A-2, you can see that the MINFREE value of 120 is constantly being reached. Therefore, page replacement occurs and in this particular case, the free list even reaches zero at one point. When that happens, threads needing free frames get blocked and cannot run until page replacement frees up some pages. To prevent this situation, you might consider increasing the MINFREE and MAXFREE values.

If you conclude that you should always have at least 1000 pages free, run the following command:

```
vmo -o minfree=1000 -o maxfree=1008
```

To make this a permanent change, include the -p flag:

```
vmo -o minfree=1000 -o maxfree=1008 -p
```

Starting with AIX 5.3, the default value of the MINFREE parameter is increased to 960 per memory pool and the default value of the MAXFREE parameter is increased to 1088 per memory pool.

Setting maxuproc

Use the following command to set the maxuproc to a value of 4096 on AIX systems.

```
chdev -l sys0 -a maxuproc='4096'
```

To check the existing value, use the following command:

```
sattr -l sys0 -E | grep maxuproc
```

The default value set by AIX (500) has been found to be too low for some large scale database environments, and causes DB2 to generate an SQL error message "`SQL1402N - Unable to authenticate user due to unexpected system error.`"

AIX asynchronous I/O (AIO) tuning for database servers

This is a setting that is best done with the help of an AIX system administrator. DB2 is a highly disk-intensive application. To improve the performance of page cleaning and prefetching, set the value of *maxservers* (usually the default is okay for minservers).

If an application does a synchronous I/O operation, it must wait for the I/O to complete. In contrast, asynchronous I/O operations run in the background and do not block user applications. This improves performance, because I/O operations and applications processing can run simultaneously. Many applications, such as databases and file servers, take advantage of the ability to overlap processing and I/O.

Applications can use the aio_read(), aio_write(), or lio_listio() subroutines (or their 64-bit counterparts) to perform asynchronous disk I/O. Control returns to the application from the subroutine as soon as the request has been queued. The application can then continue processing while the disk operation is being performed.

To manage asynchronous I/O, each asynchronous I/O request has a corresponding control block in the application's address space. This control block contains the control and status information for the request. It can be used again when the I/O operation is completed.

After issuing an asynchronous I/O request, the user application can determine when and how the I/O operation is completed. This information is provided in any of three ways:

- The application can poll the status of the I/O operation.
- The system can asynchronously notify the application when the I/O operation is done.
- The application can block until the I/O operation is complete.

Each I/O is handled by a single kproc, and typically the kproc cannot process any more requests from the queue until that I/O has completed. The default minimum number of servers configured when asynchronous I/O is enabled is one. This is the minservers attribute. There is also a maximum number of asynchronous I/O servers that can get created and which is controlled by the maxservers attribute, which has a default value of 10 per CPU. The number of servers limits the number of asynchronous disk I/O operations that can be in progress in the system simultaneously. The number of servers can be set with the SMITTY command (smitty → Devices → Asynchronous I/O → Change/Show Characteristics of Asynchronous I/O → {MINIMUM | MAXIMUM} number of servers or smitty aio) or with the **`chdev`** command.

In systems that seldom run applications that use asynchronous I/O, the defaults are usually adequate.

If the number of asynchronous I/O requests is high, then the recommendation is to increase maxservers to approximately the number of simultaneous I/Os there might be. In most cases, it is better to leave the minservers parameter at the default value because the AIO kernel extension generates additional servers if needed.

Note: AIO actions performed against a raw *logical volume* do not use kproc server processes. The setting of maxservers and minservers have no effect in this case.

By looking at the CPU utilization of the AIO servers, if the utilization is evenly divided among all of them, this means that they are all being used, and you might want to try increasing them in this case. To see the AIO servers by name, run the **`pstat -a`** command. Run the **`ps -k`** command to see the AIO servers as the name *kproc*.

For environments in which the performance of asynchronous disk I/O is critical and the volume of requests is high, but you do not have an approximate number of simultaneous I/Os, it is recommended that maxservers be set to at least 10*(number of disks accessed asynchronously).

This can be achieved for a system with three asynchronously accessed disks as follows:

```
# chdev -l aio0 -a maxservers='30'
```

In addition, you can set the maximum number of asynchronous I/O REQUESTS outstanding, and the server PRIORITY. If you have a system with a high volume of asynchronous I/O applications, it might be appropriate to increase the REQUESTS number and lower the PRIORITY number.

Viewing ibmslapd environment variables on AIX

To view the environment settings and variables for your ibmslapd process, run the following command:

```
ps ewww PID | tr ' ' '\012' | grep = | sort
```

Where PID is the ibmslapd process ID.

See the output in Example A-3.

Example: A-3 Process information for ibmslapd

```
ACLCACHE=YES
ACLCACHESIZE=25000
AIXTHREAD_SCOPE=S
AUTHSTATE=compat
A__z=!
CLASSPATH=/home/ldapdb2/sqllib/java/db2java.zip:/home/ldapdb2/sqllib/java/
   db2jcc.jar:/home/ldapdb2/sqllib/function:/home/ldapdb2/sqllib/java/
   db2jcc_license_cisuz.jar:/home/ldapdb2/sqllib/java/db2jcc_license_cu.jar:.
DB2CODEPAGE=1208
DB2INSTANCE=ldapdb2
HOME=/
IDS_LDAP_HOME=/opt/IBM/ldap/V6.0
LANG=en_US
LC__FASTMSG=true
```

```
LD_LIBRARY_PATH=/home/ldapdb2/sqllib/lib
LIBPATH=/opt/IBM/ldap/V6.0/lib64:/usr/lib:/home/ldapdb2/idsslapd-ldapdb2/
   db2instance/lib:/opt/IBM/ldap/V6.0/db2/lib64:/usr/lib:/lib:/home/ldapdb2/
   sqllib/lib:.
LOCPATH=/usr/lib/nls/loc
LOGIN=root
LOGNAME=root
MAIL=/usr/spool/mail/root
MAILMSG=[YOU
MALLOCTYPE=buckets
NLSPATH=/opt/IBM/ldap/V6.0/nls/msg/%L/%N:/opt/IBM/ldap/V6.0/nls/msg/%L/%N.cat:/
   usr/lib/nls/msg/%L/%N:/usr/lib/nls/msg/%L/%N.cat
NODISCLAIM=TRUE
ODBCCONN=15
ODMDIR=/etc/objrepos
PATH=/opt/IBM/ldap/V6.0/sbin:/opt/IBM/ldap/V6.0:/usr/bin:/etc:/usr/sbin:/usr/
   ucb:/usr/bin/X11:/sbin:/usr/java14/jre/bin:/usr/java14/bin:/usr/java131/jre/
   bin:/usr/java131/bin:/home/ldapdb2/sqllib/bin:/home/ldapdb2/sqllib/adm:/
   home/ldapdb2/sqllib/misc
PWD=/home/ldapdb2/idsslapd-ldapdb2/workdir
RDBM_CACHE_BYPASS_LIMIT=100
RDBM_CACHE_SIZE=25000
RDBM_FCACHE_SIZE=25000
SHELL=/usr/bin/ksh
SSH_CLIENT=9.48.85.122
SSH_CONNECTION=9.48.85.122
SSH_TTY=/dev/pts/1
TERM=xterm
TISDIR=/opt/IBM/ldap/V6.0
TZ=CST6CDT
USER=root
VWSPATH=/home/ldapdb2/sqllib
_=/opt/IBM/ldap/V6.0/sbin/64/ibmslapd
instname=ldapdb2
location=/home/ldapdb2
```

B

How to apply DB2 fix packs to an LDAP server

Tivoli Directory Server requires DB2 to be at its latest level (at least Fix Pack 9 or later of DB2 8.1) to take advantage of the performance fixes.

http://www.ibm.com/software/data/db2/udb/support/

The latest fix pack level at the time of writing this IBM Redpaper is Version 12.

© Copyright IBM Corp. 2007. All rights reserved.

Prerequisites

To determine if the required prerequisites are installed, issue the appropriate command for your operating system to display information about the currently installed version of DB2.

Consult the following Web sites for the latest software, hardware, operating system, and product fix information:

- For operating system requirements, see:

  `http://www.ibm.com/software/data/db2/udb/sysreqs.html`

- For a list of all fixed bugs (called APARs) and support news, see:

  `http://www.ibm.com/software/data/db2/udb/support.html`

- For UNIX:

  Table B-1 shows the commands for each Linux and UNIX operating system.

Table B-1 DB2 status calls

Operating system	Command	Output to look for
IBM AIX	lslpp -al “db2_08_01.client*”	db2_08_01.client 8.1.0.0 or later (for AIX 4.3.3) db2_08_01.client 8.1.1.0 or later (for AIX 5) Sample outputs for DB2 for AIX 4.3.3: ► db2_08_01.client 8.1.0.0 COMMITTED ... ► 8.1.0.3 COMMITTED ... ► 8.1.0.8 COMMITTED ... Check the largest installation signature that is returned (8.1.0.x) to make sure it is smaller than the VRMF of the current DB2 level. For AIX 5, the signature looks like 8.1.1.y.
HP-UX	swlist -l product “*DB2*”	► DB2V8CAE 8.1.0.x [product name] ► PDB2... 8.1.0.x Product Patch Where x must be smaller than the current level (the fourth digit in VRMF).
Linux	rpm -qa l grep db2	► db2cliv81-8.1.0-x ► db2cliv81-8.1.1-x on Linux/AMD64 Where x must be smaller than the current level (the fourth digit in VRMF).
Sun Solaris operating environment	pkginfo -l db2cliv81 l grep VERSION	► VERSION: 8.1.0.x Where x must be smaller than the current level (the fourth digit in VRMF).

- For Windows:

  Starting with DB2 Universal Database (UDB) Version 8 for Windows Fix Pack 3, IBM is providing product-specific fix packs instead of one general fix pack. This change affects only DB2 Version 8 products on Windows platforms.

  This new delivery mechanism provides the following advantages:

  – Less disk space is required on the operating system drive.

  – The overall amount of disk space required for the installation is reduced.

  – Easier installation for rollout scenarios at a fix pack or modification level later than Version 8 general availability (GA) level.

  If you have received special fixes from IBM support, you must contact IBM support before you install DB2 UDB Version 8.2 Fix Pack 4 and later to check if you require an updated version of the special fixes. This helps to ensure that your system is in a consistent state and that no special fixes are lost.

You must have a DB2 product Version 8 installed at a lower VRMF (version, release, modification/maintenance level, and fix) level than DB2 Version 8.2 Fix Pack 4 and later before you install Version 8.2 Fix Pack 4 and later.

If you have a DB2 Version 8 product already installed, it must be at a lower service level than DB2 Version 8.2 Fix Pack 4 and later. There are various methods to determine the version and level of the currently installed DB2 product. You can:

- Enter the **`db2level`** command from a command prompt. The command displays the informational tokens, for example "DB2 v8.1.5.xxx" indicates Fix Pack 5 is installed.
- Open the **Control panel** and select **Add/Remove Programs**. Look for entries starting with DB2. These are the DB2 products installed on your computer. Click the support information link to display the version installed. For example, Version 8.1.3 indicates Fix Pack 3 is installed.
- Enter the **`regedit`** command from a command prompt and check the following values under the registry key:

  ```
  HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\<product name>\CurrentVersion\Version
  HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\<product name>\CurrentVersion\Release
  HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\<product
  name>\CurrentVersion\Modification
  ```

  The following values should be displayed:

  ```
  Version "8"
  Release "1"
  Modification "x"
  ```

  Where x should be lower than the level of the DB2 Version 8.2 Fix Pack 4 and later.

  The values for version, release, and modification correspond to the first three numbers of the VRMF number. The value for modification in the registry for the currently installed products must be less than the third number of the VRMF number for DB2 Version 8.2 Fix Pack 4 and later, which is 11.

Stopping all DB2 processes

We have to look at the UNIX and Windows platform separately.

- For UNIX:

  Before starting the installation, ensure that all DB2 processes are stopped.

  a. Switch to root authority by running the **`su - root`** command.

  b. Run the following commands for each instance:

  ```
  su - iname  (Where iname represents the instance owner name)
  . $HOME/sqllib/db2profile
  db2 force applications all
  db2 terminate
  db2stop
  db2licd -end      # run at each physical node
  exit
  ```

Note: If you are a High-Availability Cluster Multi-Processing (HACMP™) user, you must use the **ha_db2stop** command to stop DB2 instead of the **db2stop** command. Otherwise, the **db2stop** command triggers a failure event.

c. Run the following commands (for LDAP instance, you most likely will not have a DAS):

```
su - aname  (Where aname represents the DAS owner name)
        . $HOME/das/dasprofile
        db2admin stop
     exit
```

d. On AIX, you should also run **slibclean** to unload unused shared libraries from memory before installation.

```
/usr/sbin/slibclean
```

- For Windows:

  To stop all instances and DB2 services, use the services control panel applet (**Control Panel** → **Administrative Tools** → **Services**). If you have active database clients, you might have to force these clients off while stopping the instance. To force clients, issue the following command:

  ```
  db2stop force
  ```

Unpacking fix pack to server

We have to look at the UNIX and Windows platform separately.

- For UNIX:

  When you download and untar a fix pack or a modification level, make sure that there are no spaces in the directory path where the file was located. If there are spaces in the directory path, the installation will fail.

  For example, make sure your directory path resembles the following:

  ```
  /home/DB2FixPak/FP12/ ...
  ```

  It should *not* resemble the following:

  ```
  /home/DB2 FixPak/FP12/ ...
  ```

  Ensure sufficient file system free space (IBM AIX 4.3.3, IBM AIX 5L™, Linux, and Solaris operating environments).

  In addition to the software disk requirements, you need to have a file system with 2 GB of free space to contain the tar.Z file or tar.gz file, and the uncompressed installation image.

  Some fix pack installation images on the FTP site or the fix pack CD are in compressed or gzipped format. Before you can apply the DB2 fix pack from these formats, you have to copy the image to a temporary directory and uncompress or gunzip the fix pack installation image.

  The compressed or gzipped images might have the file name FP11_$PTF.tar.Z or FP12_$PTF.tar.gz, where FP12_$PTF represents the latest fix pack operating system name and version.

  To uncompress the fix pack installation images, perform the following steps:

  a. Copy the compressed or gzipped image to a temporary file system containing at least 2 GB of free space.

b. Change to the directory where you copied the image by entering cd /TMP, where /TMP represents the directory where you copied the compressed image.

c. If the product has the *.tar.Z extension, enter the following command:

```
zcat <filename>.tar.Z | tar -xvf -
```

Where <filename> is the DB2 Fix pack you are applying.

d. If the product has the *.tar.gz extension, enter the following command to uncompress:

```
gunzip -c <filename>.tar.gz | tar -xvf -
```

Where <filename> is the DB2 fix pack you are applying.

Note: gunzip is part of the AIX 5L default installation setup. If you do not have gunzip, install the rpm.rte fileset from the AIX 5L installation media. The rpm.rte fileset contains gunzip. You can also download gzip for AIX5L from the following Web site:

http://www.ibm.com/servers/aix/products/aixos/linux/rpmgroups.html

e. If the compressed fix pack installation images are on a fix pack CD, there might be additional CDs with the file name extra.tar.Z or extra.tar.gz. Repeat these steps for each CD.

- For Windows:

  Before starting the installation, you must:

  – Verify that you have enough space to install the fix pack or the modification level. You need enough space to uncompress the self-extracting zip file.

  – Ensure that your system has met all of the installation prerequisites including operating system patches. This prevents technical problems that might occur after the installation and configuration of DB2 products.

  To uncompress the self-extracting image, double-click the self-extracting .exe file. For example, if you have the DB2 Enterprise Server Edition product on Windows 32-bit double click **FP12_WR21365_ESE.exe**. The WinZip Self Extractor window opens.

  You can also uncompress the fix pack installation image using an unzip-compatible utility. For example, to uncompress the DB2 Enterprise Server Edition product to a directory of your choice, enter:

  ```
  winzip32 FP12_WR21365_ESE.exe -e
  ```

  Select the folder to extract the files. Click **Unzip**. All files are extracted to the specified folder.

Installing fix pack

We have to look at the UNIX and Windows platform separately.

- For UNIX:

  To install DB2 Version 8.2 Fix Pack 4 (DB2 Version 8.1 Fix Pack 12):

  – You must be logged on as root.

  – Change to the directory in which the installation image is located.

  – To launch the installation, enter:

  ```
  ./installFixPak -y
  ```

Where the -y option indicates your agreement to the license terms and conditions. The -y option must be specified for installation to continue.

Note: By default, the installFixPak command commits all of the updated filesets on AIX.

On AIX, if you do not want to commit the updates, you should issue the installFixPak command with the -a option (for *apply* as opposed to *commit*) as follows:

```
./installFixPak -y -a
```

- For Windows:

  To install the DB2 Version 8.2 Fix Pack 4 and later on a system with a single DB2 product installed:

  Change to the folder where the unzipped files are located. The setup command is located under the folder with the product abbreviation. For example, DB2 Enterprise Server Edition would be under ESE.

  To start the DB2 Setup wizard, double-click the **setup.exe** file. The DB2 Setup wizard graphical user interface (GUI) Launchpad opens.

  Online help is available to guide you through the Setup wizard GUI. To invoke the online help, click **Help** or press F1.

Post-installation

After installing DB2 Version 8.2 Fix Pack 4 or later, do the following:

1. Update instances to use the new level of DB2.
2. Update the system catalogs.
3. Restart the instances, and bind the bind files.

Updating instances to use the new level of DB2

We have to look at the UNIX and Windows platform separately.

- For UNIX:
  - This task is mandatory. All instances must be updated after a new level of DB2 is installed.
  - *Prerequisite*: You have to be logged on as root to update the instances.
  - For each instance, issue the command:

    ```
    INSTHOME/instance/db2iupdt iname
    ```

    Where iname represents the instance name and INSTHOME represents the installation directory appropriate to your operating system.
- For Windows:

  This task is strongly recommended if you want to use capabilities specific to the latest fix pack. If you are not planning to use capabilities specific to the latest fix pack and might possibly decide to return to an earlier fix pack, you should not use **`db2updv8`**.

After installing DB2 UDB Version 8.2 Fix Pack 4 and later, run the **db2updv8** command to update the system catalogs to support the current level by enabling several built-in routines. Running the **db2updv8** command is not required, but some functionality in DB2 UDB Version 8.2 will not work if this command is not run.

After you have run the **db2updv8** command to update the system catalogs to the current Version 8 level, falling back to Version 8.1 is not supported.

For more information about the **db2updv8** command, run this command with the -h option. For technical information, search the Information Center for Update Database to Version 8 Current Level Command.

Steps to perform after applying the fix pack

We have to look at the UNIX and Windows platform separately.

- For UNIX:

  Make sure that LDAP is down when you do the following:

```
cd /home/idsldap/sqllib/bnd
db2 connect to idsldap
db2 bind @db2ubind.lst BLOCKING ALL GRANT PUBLIC ACTION ADD
db2 bind @db2cli.lst BLOCKING ALL GRANT PUBLIC
db2 bind db2schema.bnd BLOCKING ALL GRANT PUBLIC sqlerror continue
db2 bind db2clipk.bnd collection NULLIDR1
db2 bind db2clipk.bnd collection NULLIDRA
```

  We have a script that does the above commands for you on UNIX called **db2perf.sh**.

```
Usage:  db2perf.sh [ -db dbname ]
Options:
   -db dbname DB name to update with bind scripts (Default=ldapdb2)
```

  Note: This must be executed as the DB2 instance owner.

  Continue by editing the **db2cli.ini** and adding the REOPT=3 variable.

```
cd /home/idsldap/sqllib/cfg
vi db2cli.ini
```

  Add to end of this file after a “blank” line and then put a blank line after this entry. The lines should be as follows. The example we are using is [LDAPDB2B]. This is the db2 instance alias name that you can get from the ibmslapd.conf file. Make sure that this name matches the instance alias name:

```
[IDSLDAPB]
REOPT=3
```

- For Windows:

  After applying fixes, you must issue one of the following command sequences for the DB2 command line:

```
db2 terminate
db2 CONNECT TO <dbname>
db2 BIND $DB2DIR\BND\@db2ubind.lst GRANT PUBLIC ACTION ADD
db2 bind $DB2DIR\BND\@db2cli.lst BLOCKING ALL GRANT PUBLIC
db2 bind $DB2DIR\BND\db2schema.bnd BLOCKING ALL GRANT PUBLIC sqlerror continue
db2 terminate
```

C

DB2 UDB concepts and definitions

This appendix is a brief summary of some DB2 UDB concepts and terminology required to understand the contents of this section. For a more in-depth discussion of these and other DB2 UDB terms, refer to the DB2 UDB manuals available online.

The best place to go is the DB2 information center:

`http://publib.boulder.ibm.com/infocenter/db2luw/v8//index.jsp`

Instances

An *instance*, in DB2 UDB, is a logical database manager environment where you can create and/or catalog databases and set various instance-wide configuration parameters. A database manager instance can also be defined as being similar to an image of the actual database manager environment. Furthermore, you can have several instances of the database manager product on the same database server. You can use these instances to separate the development environment from the production environment, tune the database manager to a particular environment, and protect sensitive information from a particular group of people. For a partitioned database environment, all database partitions will reside within a single instance and will share a common set of configuration parameters at the instance level.

Databases

A *database* is created within an instance. A database presents logical data as a collection of database objects (for example, tables and indexes). Each database includes a set of system catalog tables that describe the logical and physical structure of the data, configuration files containing the parameter values allocated for the database, and recovery logs. DB2 UDB allows multiple databases to be defined within a single database instance. Configuration parameters can also be set at the database level to tune various characteristics, such as memory usage and logging.

© Copyright IBM Corp. 2007. All rights reserved.

Buffer pools

A *buffer pool* is the main memory allocated in the host processor to cache table and index data pages as they are being read from disk, or being modified. The purpose of the buffer pool is to improve the system performance. Data can be accessed much faster from memory than from disk, therefore, the fewer times the database manager needs to read from or write to disk (I/O), the better the performance. Buffer pools are created by database partitions and each partition can have multiple buffer pools.

Tables

The primary database object is the *table*. A table is defined as a named data object consisting of a specific number of columns and a various number of rows. Tables are uniquely identified units of storage maintained within a DB2 tablespace. They consist of a series of logically linked blocks of storage that have been given the same name. They also have a unique structure for storing information that permits that information to be related to information in other tables. When creating a table, you can choose to have certain objects, such as indexes, stored separately from the rest of the table data. In order to do this, the table must be defined to a data-managed space (DMS) tablespace.

Tablespaces

A database is logically organized into *tablespaces*. A tablespace is a place to store tables. The tablespace is where the database is defined to use the disk storage subsystem. One method to spread a tablespace over one or more physical storage devices is to simply specify multiple containers. There are three main types of user tablespaces: regular, temporary, and long. In addition to these user-defined tablespaces, DB2 also defines separate system and catalog tablespaces. For partitioned database environments, the catalog tablespace resides on the catalog database partition.

System-managed versus database-managed tablespaces

For partitioned databases, the tablespaces can reside in node groups. During the CREATE TABLESPACE command, the containers themselves are assigned to a specific database partition in the node group, thus maintaining the *shared nothing* character of DB2 UDB. Tablespaces can be either system-managed space (SMS), or data-managed space (DMS). For an SMS tablespace, each container is a directory in the file system, and the operating system's file manager controls the storage space. For a DMS tablespace, each container is either a fixed-size pre-allocated file or a physical volume, and the database manager controls the storage space itself.

Containers

A *container* is an allocation of physical storage. It is a way to define the device that is made available for storing database objects. Containers can be assigned to file systems by specifying a directory. Such containers are identified as PATH containers and are used with SMS tablespaces. Containers can also reference files that reside within a directory. These are identified as FILE containers, and a specific size must be identified. FILE containers are only used with DMS file tablespaces. Containers can also reference raw character devices. These containers are used by DMS raw tablespaces and are identified as DEVICE containers. Note that the device must already exist on the system before the container can be used. In all cases, containers must be unique and can belong to only one tablespace.

Pages

Data is transferred to and from devices in discrete blocks called *pages* that are buffered in memory. DB2 UDB supports various page sizes including 4 KB, 8 KB, 16 KB, and 32 KB. When an application accesses data randomly, the page size determines the amount of data transferred. In other words, it corresponds to the data transfer request size to the disk array.

Page size determines the maximum length of a row, and is associated with the maximum size of a tablespace. These limits are shown in Table C-1. In all cases, DB2 UDB limits the number of data rows on a single page to 255 rows.

Table C-1 Page size limits

Page size	Maximum tablespace size	Maximum row length
4 KB	64 GB	4005 B
8 KB	128 GB	8101 B
16 KB	256 GB	16293 B
32 KB	512 GB	32677 B

Extents

An *extent* is the unit at which space is allocated within a container of a tablespace for a single tablespace object. This allocation consists of multiple pages. The size of the extent is specified when the tablespace is created. Note that when data is written to a tablespace with multiple containers, the data is striped across all containers in extent-sized blocks.

Prefetch size

The number of pages that the database manager will *prefetch* can be defined for each tablespace using the PREFETCHSIZE clause with either the CREATE TABLESPACE or ALTER TABLESPACE statements. The value specified is maintained in the PREFETCHSIZE column of the SYSCAT.TABLESPACES system catalog table.

Prefetching

Prefetching is a technique for anticipating data needs and reading ahead from storage in large blocks. By transferring data in larger blocks, fewer system resources are expended and less total time is required.

Sequential prefetches read consecutive pages into the buffer pool before they are required by DB2. List prefetches are more complex. In this case, the DB2 optimizer optimizes the retrieval of randomly located data. The amount of data being prefetched is part of what determines the amount of parallel I/O activity.

Ordinarily, the database administrator should define a prefetch value large enough to allow parallel use of all of the available containers and, therefore, all of the array's physical disks. Consider the following example:

- A tablespace is defined with a page size of 16 KB using raw DMS.
- The tablespace is defined across four containers, and each container resides on a separate logical disk, and each logical disk resides on a separate Redundant Array of Independent Disks (RAID) array.
- The extent size is defined as 16 pages (or 256 KB).
- The prefetch value is specified as 64 pages (number of containers x extent size).

Assuming a user issued a query that results in a tablespace scan, which then results in DB2 performing a prefetch operation, the following happens:

- DB2 UDB recognizes that this prefetch request for 64 pages (a megabyte) evenly spans four containers, and issues four parallel I/O requests, one against each of those containers. The request size to each container is 16 pages, or 256 KB. The AIX Logical Volume Manager divides the 256 KB request to each AIX logical volume into smaller units (128 KB is the largest), and passes them on to the array as back-to-back requests against each logical disk.
- An array receives a request for 128 KB; if the data is not in cache, four arrays operate in parallel to retrieve the data.

After receiving several of these requests, the array recognizes that these DB2 UDB prefetch requests are arriving as sequential accesses, causing the array sequential prefetch to take effect.

Page cleaners

Page cleaners write dirty pages from the buffer pool to disk, reducing the chance that agents looking for victim buffer pool slots in memory will have to incur the cost of writing dirty pages to disk. For example, if you have updated a large amount of data in a table, many data pages in the buffer pool might be updated but not written into disk storage (these pages are called dirty pages). Because agents cannot place fetched data pages into the dirty pages in the buffer pool, these dirty pages must be flushed to disk storage before their buffer pool memory can be used for other data pages.

D

DB2 UDB quick reference guide

This appendix lists some of the more important and frequently used DB2 commands. More information about IBM DB2 can be obtained from the following Web sites:

- DB2 Information Center site:

  http://publib.boulder.ibm.com/infocenter/db2help/index.jsp
- DB2 DeveloperWorks site:

  http://www.ibm.com/developerworks/db2/
- IBM Problem Support 1-800-IBM-SERV

  http://www.ibm.com/software/support/probsub.html
- Fix pack download site:

  http://www.ibm.com/software/data/db2/udb/support/downloadv8.html

© Copyright IBM Corp. 2007. All rights reserved.

DB2 command line processor (CLP)

Let us take a look at some basic command line calls:

- OS Shell: `db2 <command>`
- Interactive: `db2`
- Batch: `db2 -vtf <input-file>`

  For example: `Input-file (file)`

  ```
  connect to sample;
  create table test(c1 char(8);
  insert into c1 values('abc');
  select * from test;
  update test set c1='123';
  delete from test;
  connect reset;
  ```
- Help:

  ```
  db2 ?
  db2 \?
  ```
- SQL Message: `db2 ? <SQL Message>`

  For example:

  ```
  db2 ? SQL0805
  ```

Instance configuration

For instance configuration, use the following:

- Create: `db2icrt`
- List: `db2ilist`
- Update: `db2iuptd`
- Drop: `db2idrop`

Instance configuration keywords

The instance configuration keywords are:

- Display: `GET DBM CFG`
- Update: `UPDATE DBM CFG USING <keyword> <value>`

  For example, to update the instance IP listener configuration, issue:

  ```
  UPDATE DBM CFG USING SVCENAME 50000
  ```

DB2 registry configuration

For DB2 registry configuration, use the following:

- Help: `db2set -h`
- Display: `db2set -all`
- Set: `db2set <variable>=<value>`

  For example, to enable TCP/IP protocol issue:

  ```
  db2set DB2COMM=TCPIP
  ```

Catalog remote database

Configuration Assistance GUI: `db2ca`

```
LIST DB DIRECTORY
LIST NODE DIRECTORY
CATALOG DB <dbname> AT NODE   <node name>
CATALOG TCPIP NODE <node name> REMOTE <host ip> SERVER <ip socket port>
UNCATALOG DB <dbname> UNCATALOG NODE <node name>
```

Hint: Issue TERMINATE to refresh the directory cache.

DB2 instance start/stop

The DB2 instance start/stop commands are:

- DB2 Control Center GU: `db2cc`
- Start: `db2start`
- Stop:
  - `db2stop`
  - `db2stop force`
  - `db2_kill` (UNIX/Linux only)

A complete DB2 shutdown procedure can be accomplished as follows:

```
LIST APPLICATION
FORCE APPLICATION ALL
db2stop
```

Database commands

The database commands are:

- `CREATE DATABASE <dbname> on  <file system path>`
- `DROP DATABASE <dbname>`
- `ACTIVATE DB <dbname>`
- `DEACTIVATE DB <dbname>`

Database connection

For database connection, use:

```
CONNECT TO <dbname> [USER] <userid> [USING] <password>
```

- JDBC™ T4 String: `jdbc:db2://host:port/<dbname>`
- JDBC T2 String: `jdbc:db2:<dbname>`

Create DMS tablespace example:

```
CREATE TABLESPACE PAYROLL
MANAGED BY DATABASE USING
(DEVICE'/dev/rhdisk6' 10000,
DEVICE '/dev/rhdisk7' 10000,
DEVICE '/dev/rhdisk8' 10000)
```

Create SMS tablespace example:

```
CREATE TABLESPACE ACCOUNTING
MANAGED BY SYSTEM USING ('d:\acc_tbsp','e:\acc_tbsp', 'f:\acc_tbsp')
EXTENTSIZE 64 PREFETCHSIZE 32
CREATE TABLE SALARY.....
IN ACCOUNTING INDEX IN
ACCOUNT_IDX
```

Display database object

To display database object:

```
LIST TABLES [FOR {USER | ALL | SYSTEM | SCHEMA schema-name}] [SHOW DETAIL]
DESCRIBE {[OUTPUT] {SELECT-STATEMENT | CALL-STATEMENT}| {TABLE | INDEXES FOR
TABLE} TABLE-NAME [SHOW DETAIL]}
LIST TABLESPACES [SHOW DETAIL]
LIST TABLESPACE CONTAINERS FOR tablespace-id [SHOW DETAIL]
```

Database configuration

For database configuration, use the following:

- `GET DB CFG FOR <dbname>`
- `UPDATE DB CFG FOR <dbname> USING <keyword> <value>`

Granting database privilege

To grant database privileges:

- `GRANT BINDADD| CONNECT| CREATETAB|DBADM ON DATABASE TO USER| GROUP| PUBLIC authorization-name`
- `GRANT ALL INSERT| SELECT| UPDATE ON table-name TO USER| GROUP| PUBLIC authorization-name`
- `GRANT ALTERIN| CREATEIN| DROPIN ON SCHEMA schema-name TO USER| GROUP| PUBLIC authorization-name`

Update database statistics

To update the database statistics:

```
RUNSTATS ON TABLE table-name [USER PROFILE| statistics-options]
db2rbind -d database-name -l logfile all
REORGCHK [{UPDATE | CURRENT} STATISTICS] [ON {TABLE {USER | SYSTEM | ALL |
table-name} | SCHEMA schema-name}]
```

DB2 monitoring commands

The DB2 monitoring commands are:

- `LIST APPLICATION [SHOW DETAIL]`
- `LIST APPLICATION FOR DATABASE database-name [SHOW DETAIL]`
- `UPDATE MONITOR SWITCHES USING BUFFERPOOL on, LOCK on, SORT on, STATEMENT on, TIMESTAMP on, TABLE on, UOW on`
- `GET SNAPSHOT FOR DBM`
- `GET DBM MONITOR SWITCHES`
- `GET SNAPSHOT FOR DATABASE ON database-name`
- `LIST ACTIVE DATABASES`
- `GET SNAPSHOT FOR APPLICATION ON database-name`
- `GET SNAPSHOT FOR TABLESPACE ON database-name`
- `GET SNAPSHOT FOR BUFFERPOOL on database-name`
- `GET SNAPSHOT FOR LOCKS ON database-name`
- `GET SNAPSHOT FOR DYNAMIC SQL ON database-name`

Database recovery

Enabling point-in-time recovery: `UPDATE DB CFG FOR <dbname> USING LOGRETAIN ON`

```
BACKUP DATABASE database-name TO dir/dev
LIST HISTORY BACKUP ALL FOR DATABASE database-name
RESTORE DATABASE database-name
ROLLFORWARD DATABASE database-name
```

Troubleshooting

For troubleshooting, use the following:

- Display DB2 version and service level: `db2level`
- Display jdbc driver version: `db2jcc -version`
- Changing diagnostic level for error message log

  ```
  UPDATE DBM CFG USING DIAGLEVEL 4
  ```
- Primary error message log file location: /INSTHOME/sqllib/db2dump/db2diag.log

E

Online backup of Tivoli Directory Server

IBM Tivoli Directory Server uses the IBM DB2 relational database to store directory information. While most Tivoli Directory Server administrators might not be interested in the underlying DB2 database structure and how Tivoli Directory Server uses it, an experienced DB2 administrator might be very interested, especially when determining an overall backup and restore strategy. This appendix is intended to be sufficiently detailed to allow skilled DB2 database administrators to be able to design a backup and restore strategy for their own Tivoli Directory Server environments. This strategy can include online backup support.

Online backup is a popular feature of DB2. This feature allows a backup of a database to be made while that database is being accessed by other applications (for example, Tivoli Directory Server). Before considering a backup and restore strategy that includes online backup, be aware that performing an online backup consumes a significant amount of DB2 resources. The online procedures are intended only for users with DB2 experience.

In order for DB2 online backup to be fully supported with Tivoli Directory Server 6.0, several columns defined in previous Tivoli Directory Server databases were reduced from a 2 GB to a 1 GB maximum. While this might seem like a reduction in functionality, it has been observed that most customers have entries typically less than 24 K and only a few in the 100 K - 200 K range. As a result of this feature, customers should be able to take a backup of their Tivoli Directory Server databases without losing write capabilities.

Beginning with Tivoli Directory Server 6.0, newly created databases are defined with the reduced 1 GB size columns so that they can be defined as BLOB (1G) LOGGED instead of BLOB (2G) NOT LOGGED. This is because DB2 requires that you specify the NOT LOGGED option to create a BLOB string greater than 1 GB. For columns defined to be NOT LOGGED, DB2 does not log the BLOB column and during roll-forward recovery, the BLOB will be set to NULL. In this case, a user might not notice that data is missing until the user restores the database and discovers that the blob columns are NULL after roll-forward recovery.

In this appendix, we show how to set up the online backup capabilities in DB2 for use with Tivoli Directory Server.

Additionally, for obtainable Tivoli Directory Server 5.2 databases that have been successfully migrated to Tivoli Directory Server 6.0, this section documents DB2 commands that can be used to help users evaluate and optionally migrate their existing tables for all entries that are less than 1 GB. This migration is being done *optionally* because there is no easy way in DB2 to drop a column or reduce the size, therefore the data must be exported out of the old tables and reloaded into the new tables.

This section starts with a description of the Tivoli Directory Server 6.0 database and tablespace definitions. Individual sections describe alternative Tivoli Directory Server backup and restore procedures including DB2 offline/online backup, DB2 offline restore, and redirected restore. For customers migrating from Tivoli Directory Server V5.2 to 6.0, the last section provides example documentation, which can be used to optionally evaluate and migrate Tivoli Directory Server 5.2 customer databases so that online backup can be supported.

DB2 information

DB2 backup and restore procedures are described in detail in the *DB2 Administration Guides*:

- *IBM DB2 Universal Database Administration Guide: Planning Version 8*, SC09-4822
- *IBM DB2 Universal Database Administration Guide: Implementation Version 8*, SC09-4820
- *IBM DB2 Universal Database Administration Guide: Performance Version 8*, SC09-4821

The commands referenced by the procedures are documented in the *IBM DB2 Universal Database Command Reference Version 8*, SC09-4828. The *Administration Guide* and the *Command Reference* are part of the online library installed with DB2 and the Tivoli Directory Server. They can also be obtained at the following Web site:

http://www.ibm.com/software/data/db2/library

Additional DB2 information can also be obtained using the following resource:

IBM Advanced DBA Certification Guide and Reference for DB2 Universal Database v8 for Linux, UNIX, and Windows by Dwaine R. Snow and Thomas X. Phan

Directory schema and database definitions

Tivoli Directory Server V6.0 uses directory schema files to define the underlying DB2 directory database, which is used to store the data. Preparation for recovery of Tivoli Directory Server requires backing up the files containing the Tivoli Directory Server directory configuration and schema and the DB2 databases.

Tivoli Directory Server V6.0 directory schema

The Tivoli Directory Server maintains its schema files by default in the directory server instance owner's home directory under the etc subdirectory. For example, for the ldapdb2 instance owner, the schema file location is: /home/ldapdb2/idsslapd-ldapdb2/etc.

You can also specify a different location for the schema files during instance creation.

Each time you start the server, it checks the schema files, validates them against the underlying DB2 database, and checks that the database is correctly configured to support the schema.

A new Tivoli Directory Server can be configured to have the same schema by copying the schema files to the new server instance owner's home/etc directory.

For example, to back up the schema files on AIX, where ldapdb2 is the Tivoli Directory Server instance being used and /safeplace/etc is the location where the schema files are saved, issue the following command:

```
cp /home/ldapdb2/idsslapd-ldapdb2/etc/* /safeplace/etc
```

To set up a new Tivoli Directory Server with the same schema, issue the following command:

```
cp /safeplace/etc/* /home/ldapdb2/idsslapd-ldapdb2/etc
```

Tivoli Directory Server V6.0 directory database definitions

Because DB2 backup and restore can be done at the database level, the tablespace level, or both these levels, it is important to understand the underlying structure to determine which backup and restore method might be best for different Tivoli Directory Server environments. In general, it is strongly recommended that users do *not* do DB2 backup and restore at the tablespace level for reasons detailed below.

In the examples in this paper, ldapdb2 is used as the database name. You can use the **`db2 list database directory`** and **`db2 list tablespace show detail`** commands to find the database and tablespace information for your environment.

Tivoli Directory Server directory database and tablespaces

When Tivoli Directory Server creates a database for the directory, it uses the **`db2 create database`** command to create the database. Tivoli Directory Server creates this database with four *system-managed space* (SMS) tablespaces.

You can view the tablespaces by using the following DB2 commands run under the context of the DB2 instance owner. In this paper, the ldapdb2 user is used:

```
db2 connect to <databaseName>
db2 list tablespaces
```

Example E-1 shows the tablespace output for the Tivoli Directory Server directory database on an AIX, Linux, Solaris, or HP-UX system:

Example: E-1 Example tablespace output

```
Tablespaces for Current Database
Tablespace ID                      = 0
 Name                               = SYSCATSPACE
 Type                               = System managed space
 Contents                           = Any data
 State                              = 0x0000
   Detailed explanation:
     Normal

Tablespace ID                       = 1
 Name                               = TEMPSPACE1
 Type                               = System managed space
 Contents                           = Temporary data
 State                              = 0x0000
   Detailed explanation:
     Normal
```

```
Tablespace ID                        = 2
 Name                                = USERSPACE1
 Type                                = System managed space
 Contents                            = Any data
 State                               = 0x0000
   Detailed explanation:
     Normal

Tablespace ID                        = 3
 Name                                = LDAPSPACE
 Type                                = System managed space
 Contents                            = Any data
 State                               = 0x0000
   Detailed explanation:
     Normal
```

Tivoli Directory Server data is stored in two separate tablespaces: USERSPACE1 and LDAPSPACE. By default, there is only one container or directory for each tablespace. To view the details about the USERSPACE1 tablespace, enter a DB2 command similar to the following:

```
db2 list tablespace containers for 2
```

Example output for the server instance ldapdb2

An example output for the server instance ldapdb2 is as follows:

```
Tablespace Containers for Tablespace 2
Container ID      = 0
 Name              = /home/ldapdb2/ldapdb2/NODE0000/SQL00001/SQLT0002.0
 Type              = Path
```

The default container or directory that DB2 uses for tablespace 2 (USERSPACE1) is /home/ldapdb2/ldapdb2/SQL00001/SQLT0002.0.

It contains all of the ldapdb2 database tables, which fit in a 4 K page size. This includes the attribute tables that are used for fast DB2 lookups. Tablespace 3 (LDAPSPACE) contains the remainder of the database tables, which require a 32 K page size. This includes the ldap_entry table, which contains the majority of the Tivoli Directory Server directory data and the replication tables. To view the tablespace container information for the LDAPSPACE tablespace, enter a DB2 command similar to the following:

```
db2 list tablespace containers for 3
```

Example output is as follows:

```
Tablespace Containers for Tablespace 3
Container ID             = 0
 Name                     = /home/ldapdb2/ldap32kcont_ldapdb2
 Type                     = Path
```

It is important to notice that the Tivoli Directory Server data is spread out between tablespace 2 and 3 and that both tablespaces have to be accessed for most single Tivoli Directory Server operations. For a search, the attribute tables in tablespace 2 are used to find the entries that match, but the entry information is actually returned from the ldap_entry table in tablespace 3. For an update, the attribute tables in tablespace 2 must be updated, as well as the ldap_entry (and possibly the replication tables) in tablespace 3. For this reason, it is recommended that users do backup and restore *only* at the database level so that related sets of data are kept together. If related sets of data are not kept together, recovering to a point in time where all of the data is consistent is not likely.

Tivoli Directory Server change log database and tablespaces

Tivoli Directory Server 6.0 has a function called *change log* that causes all updates to the directory to be recorded in a separate change log DB2 database (that is, a different database from the one used to hold the Tivoli Directory Server directory information tree). The change log database can be used by other applications to query and track LDAP updates. The change log function is disabled by default. The change log function should be configured only if needed, because it reduces update performance due to the additional logging overhead.

One way to check for existence of the change log function is to look for the suffix CN=CHANGELOG. If it exists, the change log function is enabled.

When Tivoli Directory Server creates a database for the change log, it uses the **`db2 create database`** command to create a database named *ldapclog*. IBM Tivoli Directory Server creates this database with four system-managed space (SMS) tablespaces identical to the ldapdb2 database described above.

You can view the tablespaces by using the following DB2 commands run under the context of the DB2 instance owner (the ldapdb2 user in these examples):

```
db2 connect to ldapclog
db2 list tablespaces
```

It is important to notice that the Tivoli Directory Server Directory information is stored in a different database (ldapdb2) from the change log database (ldapclog). In order to keep related sets of data together, care must be taken to make sure that they are backed up and restored in a consistent manner.

Distributing databases across multiple physical disks

In this section, we show how you can use DB2 offline backup and redirected restore to distribute the Tivoli Directory Server database across multiple disks.

In some cases, the Tivoli Directory Server or DB2 administrator may have altered the Tivoli Directory Server database layout by performing a redirected restore. In this case, the data might already have been redistributed, and the database layout can be different. The commands for backing up and restoring a DB2 database are the same whether or not the database has been distributed across multiple physical disks.

As the database grows, it might become necessary and desirable to distribute the database across multiple physical disk drives. You can achieve better performance by spreading entries across multiple disks. In terms of performance, one 20 GB disk is not as good as two 10 GB disks. The following sections describe how to configure DB2 to distribute the ldapdb2 database across multiple disks. Similar instructions can be followed to distribute the change log database across multiple disks. Replace the ldapdb2 database with the ldapclog database in the examples shown below.

Creating file systems and directories on the target disks

The first step in distributing the DB2 database across multiple disk drives is to create and format the file systems and directories on the physical disks that the database is to be distributed among.

The guidelines are as follows:

- Because DB2 distributes the database equally across all directories, it is a good idea to make all of the file systems, directories, or both, the same size.
- All directories to be used for the DB2 database must be completely empty. AIX and Solaris systems create a lost+found directory at the root of any file system. Instead of deleting the lost+found directory, create a subdirectory at the root of each file system to be used for distributing the database. For example, create a subdirectory named disk*n* for each file system where the DB2 database is to be stored (for example, disk1, disk2, disk3, and so on).
- Create two additional directories under the disk*n* directory: One for holding tablespace 2 and the other for tablespace 3. For example, these directories might be named ts2 and ts3. Then specify these directories on the **set tablespace** commands as discussed in "Performing a redirected restore of the database" on page 149.
- The DB2 instance user must have *write* permission on the created directories. For AIX and Solaris systems, the following command gives the proper permissions:

  ```
  chown ldapdb2 directory_name
  ```

Platform-specific guidelines

The platform-specific guidelines are:

- For the AIX operating system, create the file system as an Enhanced Journaled File System. If the file system is created as a Journaled File System, it must be defined with the Large File Enabled option. This option can be found through the Add a Journaled File System option of the smit menu.
- For AIX and Solaris systems, set the file size limit to unlimited or to a size large enough to allow for the creation of a file as large as the file system. On AIX systems, the /etc/security/limits file controls system limits and -1 means unlimited. On Solaris systems, the **ulimit** command controls system limits.

Backing up the existing database

To back up the existing database, perform these steps:

1. Stop the IBM Directory server process (ibmslapd).
2. To close all DB2 connections, enter the following:

   ```
   db2 force applications all
   db2 list applications
   ```

   A message similar to the following is displayed:

   ```
   SQL1611W No data was returned by Database System Monitor.
   ```

3. To initiate the offline backup process, enter the following:

   ```
   db2 backup db ldapdb2 to [file system | tape device]
   ```

   When the database has been backed up successfully, a message similar to the following is displayed:

   ```
   Backup successful. The timestamp for this backup image is: 20000420204056
   ```

In the message above, the timestamp value is a unique identifier for each database backup image and will be required when you perform a restore if you have more than one backup image in a single folder. On AIX, Linux, Solaris, and HP-UX systems, the timestamp is concatenated to the backup image file name:

```
DBNAME.type.instance.NODExxxx.CATNxxxx.yyyymmddhhmmss.seq
```

For example:

```
LDAPDB2.0.ldapdb2.NODE0000.CATN0000.20050404215244.001
```

On Windows systems, the backup image is stored in a 5 level directory tree, which also contains the timestamp information:

```
"Dbname.type\db2instance\nodexxx\catnxxxx\yyyymmdd\hhmmss.seq"
```

Note: Ensure that the backup process was successful before proceeding. The next step destroys the existing database in order to re-create it. If the backup was not successful, the existing database is lost. You can verify the success of the backup by restoring to a separate system. You can also use the **db2ckbkp** command to test the integrity of a backup image and to determine whether the image can be restored. For example, using the **db2ckbkp** command on the AIX, Linux, Solaris, or HP-UX backup image generated above:

```
db2ckbkp LDAPDB2.0.ldapdb2.NODE0000.CATN0000.20050404215244.001
```

```
[1] Buffers processed:
#################################################################################
#################################################################################
#################################################################################
############################################################
Image Verification Complete - successful.
```

Performing a redirected restore of the database

A DB2 redirected restore restores the specified database tablespace to multiple containers or directories. In the following example, assume that the following directories for tablespace 2 were created, are empty, and have the correct permissions to allow write access by the DB2 instance owner (the ldapdb2 user in our examples):

- /disk1/ts2
- /disk2/ts2
- /disk3/ts2
- /disk4/ts2
- /disk5/ts2

In the following example, assume that the following directories for tablespace 3 were created:

- /disk1/ts3
- /disk2/ts3
- /disk3/ts3
- /disk4/ts3
- /disk5/ts3

Follow these steps for a redirected restore:

1. To start the DB2 restore process, enter the following:

   ```
   db2 restore db ldapdb2 from location_of_backup replace existing redirect
   ```

   Messages similar to the following are displayed:

   ```
   SQL2539W Warning! Restoring to an existing database that is the same as
   the backup image database. The database files will be deleted.


   SQL1277N Restore has detected that one or more tablespace containers are
   inAccessible, or has set their state to 'storage must be defined'.


   DB20000I The RESTORE DATABASE command completed successfully.
   ```

   The message SQL1277N indicates that *storage must be defined* for the tablespace containers, which is illustrated in the following step.

2. To define the containers for tablespace 2 and for tablespace 3, enter the following:

   ```
   db2 "set tablespace containers for 2 using (path \
      '/disk1/ts2', path '/disk2/ts2', path '/disk3/ts2', \
      path '/disk4/ts2', path '/disk5/ts2')"
      db2 "set tablespace containers for 3 using (path \
      '/disk1/ts3', path '/disk2/ts3', path '/disk3/ts3', \
      path '/disk4/ts3', path '/disk5/ts3')"
   ```

   Note: If many containers are defined, these commands can become so long that they do not fit within the limits of a shell command. In this case, you can put the command in a file and run within the current shell using the dot notation. For example, assume that the commands are in a file named set_containers.sh. The following command runs the set_containers.sh commands in the current shell:

   ```
   . set_containers.sh
   ```

   After completion of the DB2 **`set tablespace containers`** command, a message similar to the following is displayed:

   ```
   DB20000I The SET TABLESPACE CONTAINERS command completed successfully.
   ```

   If you receive the message `SQL0298N  Bad container path. SQLSTATE=428B2`, it indicates that one of the containers is not empty or that *write* permission is not enabled for the DB2 instance owner (the ldapdb2 user in these examples).

   Note: A newly created file system on AIX and Solaris contains a directory named lost+found. You should create a directory at the same level as lost+found to hold the tablespace and then re-issue the set tablespace command. If you experience problems, see the DB2 documentation. The following files might also be of interest:

   - ldapdb2_home_dir /sqllib/Readme/en_US/Release.Notes
   - ldapdb2_home_dir /sqllib/db2dump/db2diag.log

   Note that the db2diag.log file contains some fairly low-level details that can be difficult to interpret. In many cases, the information can be generally used to determine if and what type of error occurred.

3. Continue the restore to new tablespace containers. This step, which restores the entire database from its old containers to the new containers, takes the most time to complete.

The time varies depending on the size of the directory. To continue the restore to the new tablespace containers, enter the following:

```
db2 restore db ldapdb2 continue
```

If successful, the following message is displayed:

```
DB20000I The RESTORE DATABASE command completed successfully.
```

If problems occur with the redirected restore and you want to restart the restore process, it might be necessary to enter the following command first:

```
db2 restore db ldapdb2 abort
```

If the redirected restore has been stopped, the redirected restore can be restarted, beginning at step 1.

Overview of backup and restore procedures for LDAP

The fastest way to back up and restore the database is to use DB2 **backup** and **restore** commands. Tivoli Directory Server alternatives, such as **db2ldif** and **ldif2db**, are generally much slower in comparison. There are also the Tivoli Directory Server supported tools **idsdbback** and **idsdbrestore**, which use the DB2 **backup** and **restore** commands and also save additional Tivoli Directory Server configuration and schema information. However, it is important to note that **idsdbback** does not support DB2 online backup in Tivoli Directory Server 6.0. You can use **idsbback** only when the Tivoli Directory Server is *not* running.

A disadvantage of using the **db2 backup** and **db2 restore** commands is that the backed up database cannot be restored across dissimilar hardware platforms. For example, you cannot back up an AIX database and restore the database to a Solaris system. You also cannot back up a database on one version of Tivoli Directory Server and then restore that database on another version of Tivoli Directory Server. You should use the same version of DB2 for both the **db2 backup** and **db2 restore** operations.

As an alternative to the **db2 backup** and **db2 restore** commands, you can use the LDAP Data Interchange Format (LDIF) export and import commands: **db2ldif** (export) and **ldif2db** (import). These commands work across dissimilar hardware platforms, but the process is slower.

An important advantage of using **db2 backup** and **db2 restore** commands or the **dbback** and **dbrestore** commands is the preservation of DB2 configuration parameters and database optimizations in the backed up database. The restored database has the same performance tuning tasks as the backed up database. This is not the case with the Tivoli Directory Server **db2ldif** and **ldif2db** commands.

Be aware that if you restore over an existing database, any performance tuning tasks on that existing database are lost. Check all DB2 configuration parameters after performing a restore. Also, if you do not know whether a **db2 reorgchk** was performed before the database was backed up, run **db2 reorgchk** after the restore.

The DB2 commands to perform offline backup and restore operations for a directory database named **ldapdb2** are as follows:

```
db2 force applications all
db2 backup db ldapdb2 to directory_or_device
db2 restore db ldapdb2 from directory_or_device replace existing
```

Where directory_or_device is the name of a directory or device where the backup is stored.

The DB2 commands to perform offline backup and restore operations for the change log database are as follows:

```
db2 force applications all
db2 backup db ldapclog to directory_or_device
db2 restore db ldapclog from directory_or_device replace existing
```

The most common error that occurs on a restore is a file permission error. Some reasons why this error might occur are:

- The DB2 instance owner does not have permission to access the specified directory and file. One way to solve this is to change directory and file ownership to the DB2 instance owner. For example, enter the following:

  ```
  chown ldapdb2 fil_or_dev
  ```

- The backed up database is distributed across multiple directories, and those directories do not exist on the target system of the restore. Distributing the database across multiple directories is accomplished with a redirected restore. To solve this problem, either create the same directories on the target system or perform a redirected restore to specify the proper directories on the new system. If creating the same directories, ensure that the owner of the directories is the DB2 instance owner (the ldapdb2 user in these examples). For more information about redirected restore, see "Distributing databases across multiple physical disks" on page 147.

Replication considerations

Backup and restore operations can be used to initially synchronize a Tivoli Directory Server consumer with a Tivoli Directory Server supplier or whenever the supplier and consumer get out of sync. A consumer can get out of sync if it is not defined to the supplier. In this case, the supplier does not know about the consumer and does not save updates on a propagation queue for that consumer.

Overview of online backup and restore procedures for LDAP

When the Tivoli Directory Server database is first created, only circular logging is enabled for it. This means that log files are reused (in a circular fashion), and are not saved or archived. With circular logging, roll-forward recovery is not possible; only crash recovery is enabled. The directory server must be stopped and offline when backups are taken.

When log archiving is configured for the database, roll-forward recovery is possible. This is because the logs record changes to the database after the time that the backup was taken. You perform log archiving by having the logretain database configuration parameter set to RECOVERY. When this parameter is configured, the database is enabled for roll-forward recovery.

After logretain is set to RECOVERY, a full offline backup of the database must be made for the *backup pending state* to be satisfied so that the database can be used. To check if the database is in backup pending state, look at the Backup pending value (YES or NO) returned from the following DB2 command:

```
db2 get db config for ldapdb2
```

When the database is recoverable, the backups of the database can be completed online. Roll-forward recovery reapplies the completed units of work recorded in the logs to the restored database, tablespace, or tablespaces. You can specify that roll-forward recovery is either to the end of the logs, or to a particular point in time.

A recovery history file is created with each database. The recovery history file is updated automatically with summary information whenever you perform a backup or restore of a full database or tablespace. This file can be a useful tracking mechanism for restore activity within a database. This file is created in the same directory as the database configuration file. It is automatically updated whenever there is one of the following activities:

- Backup of a database or tablespace
- Restore of a database or tablespace
- Roll-forward of a database or tablespace
- Alter of a tablespace
- Quiesce of a tablespace
- Rename of a tablespace
- Load of a table
- Drop of a table
- Reorganization of a table
- Update of table statistics

For information about previously backed up databases, enter the following DB2 command:

```
db2 list history backup all for db ldapdb2
```

The database configuration file contains the logretain and other parameters related to roll-forward recovery. Because in some cases, the default parameter settings does not work well, you might have to change some of these defaults for your setup. See the *DB2 Administration Guide* for detailed information about configuring these parameters in DB2.

- Primary logs (logprimary)

  This parameter specifies the number of primary logs that are created.

- Secondary logs (logsecond)

  This parameter specifies the number of secondary log files that are created and used for recovery log files (only as needed).

- Log size (logfilsiz)

  This parameter determines the number of pages for each of the configured logs. A page is 4 KB in size.

- Log Buffer (logbufsz)

  This parameter enables you to specify the amount of database shared memory to use as a buffer for log records before writing these records to disk.

- Number of Commits to Group (mincommit)

  This parameter enables you to delay the writing of log records to disk until a minimum number of commits have been performed.

- New log path (newlogpath)

  You can change the location where active logs and future archive logs are placed by changing the value for this configuration parameter to point to either a different directory or a device.

- Log retain (logretain)

  This parameter causes archived logs to be kept in the database log path directory. Enabling it by setting it to RECOVERY enables the database manager to use the roll-forward recovery method. After logretain is set to RECOVERY, you must make a full backup of the database. This state is indicated by the backup_pending flag parameter.

- Track modified pages (trackmod)

  When this parameter is set to Yes, the database manager tracks database modifications so that the backup utility can detect which subsets of the database pages must be examined by an incremental backup and potentially included in the backup image. After setting this parameter to Yes, you must take a full database backup in order to have a baseline against which incremental backups can be taken.

Basic examples for both offline and online backup of the database are described in the following sections. Example E-2 is for the AIX operating system, and might have to be modified for other operating systems.

Example DB2 list history information

In this section, we show a sample output for the DB2 backup history.

Example: E-2 Sample output for the DB2 backup history

```
db2 list history backup all for ldapdb2
                    List History File for ldapdb2
Number of matching file entries = 6
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
 -- --- ------------------ ---- --- ------------ ------------ --------------
  B  D  20050404215244001   F    D  S0000000.LOG S0000000.LOG
 ----------------------------------------------------------------------------
  Contains 3 tablespace(s):
00001 SYSCATSPACE
  00002 USERSPACE1
  00003 LDAPSPACE
 ----------------------------------------------------------------------------
 Comment: DB2 BACKUP LDAPDB2 OFFLINE
 Start Time: 20050404215244
 End Time: 20050404215324
 ----------------------------------------------------------------------------
 00001 Location: /safeplace/sun-full-ldapdb2
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
 -- --- ------------------ ---- --- ------------ ------------ --------------
  B  D  20050404215341001   N    D  S0000000.LOG S0000001.LOG
 ----------------------------------------------------------------------------
  Contains 3 tablespace(s):
00001 SYSCATSPACE
  00002 USERSPACE1
  00003 LDAPSPACE
 ----------------------------------------------------------------------------
 Comment: DB2 BACKUP LDAPDB2 ONLINE
 Start Time: 20050404215341
  End Time: 20050404215411
 ----------------------------------------------------------------------------
  00002 Location: /safeplace/no-changes
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
 -- --- ------------------ ---- --- ------------ ------------ --------------
  B  D  20050404221134001   N    D  S0000006.LOG S0000011.LOG
 ----------------------------------------------------------------------------
  Contains 3 tablespace(s):
00001 SYSCATSPACE
```

```
  00002 USERSPACE1
  00003 LDAPSPACE
 ---------------------------------------------------------------------------
  Comment: DB2 BACKUP LDAPDB2 ONLINE
  Start Time: 20050404221134
  End Time: 20050404221228
 ---------------------------------------------------------------------------
  00004 Location: /safeplace/duringadd
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
 -- --- ------------------ ---- --- ------------ ------------ --------------
  B  D  20050404222613001   N    D  S0000055.LOG S0000057.LOG
 ---------------------------------------------------------------------------
  Contains 3 tablespace(s):
00001 SYSCATSPACE
  00002 USERSPACE1
  00003 LDAPSPACE
 ---------------------------------------------------------------------------
  Comment: DB2 BACKUP LDAPDB2 ONLINE
  Start Time: 20050404222613
  End Time: 20050404222706
 ---------------------------------------------------------------------------
  00005 Location: /safeplace/addsdone
Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log  Backup ID
 -- --- ------------------ ---- --- ------------ ------------ --------------
  B  D  20050405170536001   N    D  S0000058.LOG S0000059.LOG
 ---------------------------------------------------------------------------
  Contains 3 tablespace(s):
00001 SYSCATSPACE
  00002 USERSPACE1
  00003 LDAPSPACE
 ---------------------------------------------------------------------------
  Comment: DB2 BACKUP LDAPDB2 ONLINE
  Start Time: 20050405170536
  End Time: 20050405170635
 ---------------------------------------------------------------------------
  00006 Location: /safeplace/no-changes-since-yest
```

Example offline backup and restore procedures

The first set of four steps explains the backup procedure.

1. Determine a secure location to store the files to be used for backup and recovery: a backup machine, separate media, and so on. /safeplace is used for the examples below. (The DB2 instance owner must have *write* permission for the /safeplace directory.)
2. Save the Tivoli Directory Server configuration and schema files in a secure location. These files have to be updated only if you change the topology, change your configuration parameters, or change your schema. (ldapdb2 is used as the directory server instance name.)

   ```
   cp /home/ldapdb2/idsslapd-ldapdb2/etc/* /safeplace/etc
   ```
3. Make sure that ibmslapd is not running:

   ```
   ibmslapd -I ldapdb2 -k
   ```

4. Create a full database offline backup on Sunday. (Be sure to run all DB2 commands as the DB2 instance owner.)

   ```
   db2 force applications all
   db2 backup db ldapdb2 to /safeplace/sun-full-ldapdb2
   ```

The second set of five steps restores the directory database on a different machine.

1. If necessary, install Tivoli Directory Server.
2. Configure a new database, using the same information that was specified on the backup machine.
3. Copy or FTP the configuration, schema, and backup image files from the backup machine to /safeplace on this machine.
4. Copy the backed up configuration and schema files to this machine:

   ```
   cp /safeplace/etc/* /home/ldapdb2/idsslapd-ldapdb2/etc
   ```

5. Restore the directory database from Sunday:

   ```
   db2 restore db ldapdb2 from /safeplace/sun-full-ldapdb2 replace existing
   ```

Note: In some versions and cases, DB2 supports cross-platform backup and restore operations and mixed version backup and restore operations. From a Tivoli Directory Server perspective, you cannot back up a database on one version of Tivoli Directory Server and then restore that database on another version of Tivoli Directory Server. It is recommended that you use the same version of **`db2 backup`** and **`db2 restore`** for both DB2 operations.

Example online backup for the directory database

The first set of seven steps describes how full online backups are used for recovery.

1. Determine a secure location to store files to be used for backup and recovery: a backup machine, separate media, and so on. /safeplace is used for the examples below. (The DB2 instance owner must have write permission for the /safeplace directory.)
2. Save the Tivoli Directory Server configuration and schema files in a secure location. These files have to be updated only if you change the topology, change your configuration parameters, or change your schema. (ldapdb2 is used as the Tivoli Directory Server instance name.)

   ```
   cp /home/ldapdb2/idsslapd-ldapdb2/etc/* /safeplace/etc
   ```

3. Make sure that ibmslapd is not running:

   ```
   ibmslapd -I ldapdb2 -k
   ```

4. For recovery purposes, log files should be kept on a different physical drive from the database. For this example, /safeplace/db2logs-ldapdb2 is the secure location. (You must run all DB2 commands as the DB2 instance owner.)

   ```
   db2 update db config for ldapdb2 using newlogpath /safeplace/db2logs-ldapdb2
   ```

5. Update the Directory Server database for online backup support (logretain on):

   ```
   db2 update db config for ldapdb2 using logretain recovery
   db2 force applications all
   db2stop
   db2start
   ```

   After logretain is set to recovery, you must make a *full offline backup.*

6. Create a full database offline backup on Sunday:

   ```
   db2 backup db ldapdb2 to /safeplace/sun-full-ldapdb2
   ```

7. Start the directory server instance:

   ```
   ibmslapd -I ldapdb2
   ```

The second set of two steps shows you how to create nightly full online backups for the directory database.

1. On a nightly basis (or more frequently, if determined necessary), create full backups and copy log files from the log file path. Run the same commands on Tuesday, Wednesday, Thursday, Friday, and Saturday.

   Important: You can use an online backup image for recovery only if you have the logs that span the time during which the backup operation was running:

   ```
   db2 backup db ldapdb2 online to /safeplace/mon-ldapdb2
   ```

2. Verify the log path. (DB2 appends the node to the path specified.)

   ```
   db2 get db config for ldapdb2 | grep "Path to log files"
   ```

   The following is an example of the information returned:

   ```
   Path to log files        = /safeplace/db2logs-ldapdb2/NODE0000/
   ```

Restoring the directory database

Assume that a disk drive failed on Wednesday morning on the machine being used in the above example. Because /safeplace that was used to back up the files and logs was not affected, it can be used for restore.

If a different machine is being used to restore the database, the /safeplace directories on the backed up machine must be set up on the new machine to a local /safeplace directory. This must include all backup directories being used, as well as the log files in the /safeplace/db2log-ldapdb2/NODE0000 directory.

1. If necessary, install Tivoli Directory Server.
2. Configure a new database using the same information that was specified previously.
3. Copy (or tar) the configuration and schema files backed up previously:

   ```
   cp  /safeplace/etc/*  /home/ldapdb2/idsslapd-ldapdb2/etc
   ```

4. Restore the directory database from Tuesday night:

   ```
   db2 restore db ldapdb2 from /safeplace/tues-ldapdb2 taken at
   timestamp_of_backup
   ```

   Note: The *timestamp_of_backup* clause is only required if there is more than one backup image in the specified directory path.

   If you are restoring on a new machine, you will see the following warning message:

   ```
   SQL2523W  Warning!  Restoring to an existing database that is different from the
   database on the backup image, but have matching names. The target database will
   be overwritten by the backup version. The Roll-forward recovery logs associated
   with the target database will be deleted.
   ```

```
Do you want to continue ? (y/n) y
DB20000I  The RESTORE DATABASE command completed successfully.
```

5. Set the new database's log path to the same path originally used for the log files. If you are restoring on a new machine, you must copy the log files from the old system to the new:

   ```
   db2 update db config for ldapdb2 using newlogpath /safeplace/db2logs-ldapdb2
   ```

6. Roll forward all logs located in the log directory, which include changes since the Tuesday night backup:

   ```
   db2 rollforward db ldapdb2 to end of logs and stop
   ```

   Note: In this case, recovery requires only the last full backup image and the logs spanning the time since the backup was made.

Incremental directory and change log database online backup

For incremental directory and change log database online backup:

1. Determine a secure location to store files to be used for backup and recovery: a backup machine, separate media and so on. /safeplace is used for the examples below. (If the change log is not configured, all commands containing ldapclog can be ignored.)
2. Save the Tivoli Directory Server configuration and schema files in a secure location. These files have to be updated only if you change the topology, change your configuration parameters, or change your schema. ldapdb2 is used as the Tivoli Directory Server instance.

   ```
   cp /home/ldapdb2/idsslapd-ldapdb2/etc/*  /safeplace/etc
   ```

3. Make sure that ibmslapd is not running:

   ```
   ibmslapd -I ldapdb2 -k
   ```

   Note: In this example, the path to the log files has not been modified from the default locations. This is to illustrate the default log path locations when both directory and change log databases are used. For recovery purposes, log files should be kept on a different physical drive than the databases.

4. Update the directory server database and change log database for online backup support (*logretain on*) and incremental backup (*trackmod on*).

   Note: Setting trackmod on for incremental backup support can have an impact on the runtime performance for operations that update or insert data.

   ```
   db2 update db cfg for ldapdb2 using logretain recovery trackmod on
   db2 update db config for ldapclog using logretain recovery trackmod on
   db2 force applications all
   db2stop
   db2start
   ```

Creating full offline backups for directory and change log databases

To create full offline backups for directory and change log databases:

1. Create full database offline backups for both databases on Sunday.

   ```
   db2 backup db ldapdb2 to /safeplace/sun-full-ldapdb2
   db2 backup db ldapclog to /safeplace/sun-full-ldapclog
   ```

2. Start the directory server instance:

   ```
   ibmslapd -I ldapdb2
   ```

Creating incremental online backups for directory and change log databases

To create incremental online backups for directory and change log databases:

1. On a nightly basis (or more frequently, if determined necessary), create incremental backups. Run the same commands on Tuesday, Wednesday, Thursday, Friday, and Saturday.

   Important: You can only use an online backup image for recovery if you have the logs that span the time during which the backup operation was running.

   (Note that the directory and change log database logs are kept in different paths with identical names (for example, S0000000.LOG, S0000001.LOG, and so on), therefore they have to be saved in different directories *if* the change log is configured.)

   ```
   db2 backup db ldapdb2 online incremental to /safeplace/mon-ldapdb2
   ```

2. Verify the path to the log files for the directory database:

   ```
   db2 get db config for ldapdb2 | grep "Path to log files"
   ```

   The following is an example of the information returned:

   ```
   Path to log files    = /home/ldapdb2/ldapdb2/NODE0000/SQL00001/SQLOGDIR/
   cp /home/ldapdb2/ldapdb2/NODE0000/SQL00001/SQLOGDIR/*/safeplace/db2logs-ldapdb2
   db2 backup db ldapclog online incremental to /safeplace/mon-ldapclog
   ```

3. Verify the path to the log files for the change log database:

   ```
   db2 get db config for ldapclog | grep "Path to log files"
   ```

   The following is an example of the information returned:

   ```
   Path to log files   = /home/ldapdb2/ldapdb2/NODE0000/SQL00002/SQLOGDIR/
   cp /home/ldapdb2/ldapdb2/NODE0000/SQL00002/SQLOGDIR/*
   /safeplace/db2logs-ldapclog
   ```

Restoring both directory and change log databases

Assume a disk drive failed on Wednesday morning on the machine being used in the above example. Because the /safeplace directory used to back up the files was not affected, it can be used for restore.

If a different machine is being used to restore the database, the /safeplace directories on the backed up machine must be set up on the new machine to a local /safeplace directory. This must include all backup directories being used, as well as the log files in the /safeplace/db2log-ldapdb2/NODE0000 and the /safeplace/db2log-ldapclog/NODE0000 directories.

1. If necessary, install Tivoli Directory Server.
2. Configure a new database, using the same information that was specified previously.
3. Copy the configuration and schema files backed up previously:

   ```
   cp  /safeplace/etc/*  /home/ldapdb2/idsslapd-ldapdb2/etc
   ```

4. Make sure that ibmslapd is not running:

   ```
   ibmslapd -I ldapdb2 -k
   ```

5. Restore the directory database. The last backup image to be restored is called the target image. The target image must be restored twice, once at the start of the restore procedure and again at the end. Therefore, in order to restore Tuesday's incremental backup:

   ```
   db2 restore db ldapdb2 incremental from /safeplace/tues-ldapdb2
   db2 restore db ldapdb2 incremental from /safeplace/sun-full-ldapdb2
   db2 restore db ldapdb2 incremental from /safeplace/tues-ldapdb2
   ```

6. Copy the log files backed up previously to the default log path locations:

   ```
   cp /safeplace/db2logs-ldapdb2/*/home/ldapdb2/ldapdb2/NODE0000/SQL00001/SQLOGDIR
   db2 rollforward db ldapdb2 to end of logs and stop
   ```

7. Restore the change log database:

   ```
   db2 restore db ldapclog incremental from /safeplace/tues-ldapclog
   db2 restore db ldapclog incremental from /safeplace/sun-full-ldapclog
   db2 restore db ldapclog incremental from /safeplace/tues-ldapclog
   ```

8. Copy the log files backed up previously to the default log path locations:

   ```
   cp /safeplace/db2logs-ldapclog/*
   /home/ldapdb2/ldapdb2/NODE0000/SQL00002/SQLOGDIR
   db2 rollforward db ldapclog to end of logs and stop
   ```

Note: In this case, recovery requires a full backup image, and the LAST incremental backup. Note that the Monday incremental backup is not needed to restore up through Tuesday.

Using incremental delta backups

In the examples above using incremental backup, the incremental backup increases in size until the next full backup. This is because the backup contains accumulated changes over time, therefore there are many more changes saved for Saturday than there were for Monday. DB2 also allows *delta* backups, which saves only changes made since the last backup of any kind. These delta backups are much smaller and finish more quickly; however, when used to restore, you will require *all* deltas since the last full or incremental backup.

The commands to perform nightly online delta backups for the ldapdb2 database above are:

```
db2 backup db ldapdb2 online incremental delta to /safeplace/mon-delta-ldapdb2
db2 backup db ldapdb2 online incremental delta to /safeplace/tues-delta-ldapdb2
db2 backup db ldapdb2 online incremental delta to /safeplace/wed-delta-ldapdb2
db2 backup db ldapdb2 online incremental delta to /safeplace/thurs-delta-ldapdb2
db2 backup db ldapdb2 online incremental delta to /safeplace/fri-delta-ldapdb2
db2 backup db ldapdb2 online incremental delta to /safeplace/sat-delta-ldapdb2
```

As shown in the previous examples, the log files for the database must be kept in a secure place also when using delta backups. If you are using the default log paths, you must copy them to a /safeplace/db2logs-ldapdb2 directory, or modify the database configuration to save them directly in the /safeplace/db2logs-ldapdb2 location.

Restoring from incremental delta backups

As shown in the previous examples, the log files for the database from the backup machine must be available on the machine being used for restoring the delta backups. If you are using the default log paths, you must copy them from the /safeplace/db2logs-ldapdb2/NODE0000 directory on the backup machine to the default log path on the machine being restored, or modify the database configuration newlogpath on the new machine and copy them directly to the /safeplace/db2logs-ldapdb2/NODE000 location.

When restoring from delta backups, you must have *all* deltas since the last full or incremental backup.

The commands to restore online delta backups for the ldapdb2 database above are:

```
db2 restore db ldapdb2 incremental from /safeplace/sat-delta-ldapdb2
db2 restore db ldapdb2 incremental from /safeplace/sun-full-ldapdb2
db2 restore db ldapdb2 incremental from /safeplace/mon-delta-ldapdb2
db2 restore db ldapdb2 incremental from /safeplace/tues-delta-ldapdb2
db2 restore db ldapdb2 incremental from /safeplace/wed-delta-ldapdb2
db2 restore db ldapdb2 incremental from /safeplace/thurs-delta-ldapdb2
db2 restore db ldapdb2 incremental from /safeplace/fri-delta-ldapdb2
db2 restore db ldapdb2 incremental from /safeplace/sat-delta-ldapdb2
```

Note: As in the previous example, the target image must be restored twice, at the beginning and again as the last restore.Copy the logs and do the roll forward as before:

```
cp /safeplace/db2logs-ldapdb2/*
/home/ldapdb2/ldapdb2/NODE0000/SQL0001/SQLOGDIR/
db2 rollforward db ldapdb2 to end of logs and stop
```

Pros and cons of different recovery strategies

If a database has high write activity, an *online full backup* might be more efficient. Although minimal, the tracking of updates to the database can have an impact on the runtime performance of transactions that update or insert data.

Incremental backup is a good way to protect a database that is mostly read-only, but has some write activity, which makes it important to be recoverable. An incremental backup image is a copy of all database data that has changed since the most recent, successful, full backup operation. This is also known as a *cumulative backup image*, and the predecessor of an incremental backup image is always the most recent successful full backup of the same object.

An *incremental delta backup* image is a copy of all database data that has changed since the last successful backup (full, incremental, or incremental delta). This is also known as a *differential* or *non-cumulative backup image*. While delta backups are smaller, *all* deltas since the last full or incremental backup are required to restore the database.

Other backup, restore, and roll-forward command options

Use the following command in a situation where you want to restore a database to a specific point in time, and not roll forward any changes made after that point in time; the *without rolling forward* prevents DB2 from putting the restored database in roll-forward pending state:

```
db2 restore db ldapdb2 from /safeplace taken at 20040405154705 without rolling
forward
```

To restore a database from a path where there is only one backup database image stored and without prompting:

```
db2 restore db ldapclog from /safeplace/full-backup-ldapclog without rolling
forward without prompting
```

For offline roll-forward database to a point in time:

```
db2 "rollforward database ldapdb2 to 2004-04-22-14.54.21.253422 and stop
```

This command rolls forward all logs located in the log folder specified in the database configuration file up to and including the above stated point in time. The *and stop* key phrase completes the roll-forward recovery process by rolling back incomplete transactions and tuning off the roll-forward pending state of the database.

Common problems for backup, restore, and roll-forward commands

ldapdb2 is used as the database name in the examples below. For change log, the change log database (ldapclog) can be used.

- Example: Trying to update database configuration for online backup parameters while ibmslapd is running.

  ```
  db2 update db cfg for ldapdb2 using logretain recovery trackmod on
  DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.

  SQL1363W One or more of the parameters submitted for immediate modification
  were not changed dynamically. For these configuration parameters, all
  applications must disconnect from this database before the changes become
  effective.
  ```

  If you receive this message, you must stop and restart ibmslapd for the changes to take effect:

  ```
  ibmslapd -I ldapdb2 -k
  ibmslapd -I ldapdb2
  ```

- Example: Trying to perform online backup without setting logretain.

  ```
  db2 "backup database ldapdb2 online to /safeplace

  SQL2413N Online backup is not allowed because either logretain or userexit for
  roll-forward is not activated, or a backup pending condition is in effect for
  the database.
  ```

  To set the logretain parameters to enable roll-forward recovery for database ldapdb2, the following DB2 command must be run:

  ```
  db2 update db config for ldapdb2 using logretain recovery
  ```

After logretain is set to recovery, the user must make a full backup of the database. This state is indicated by the backup_pending flag parameter. If a full backup has not been made, the following message is displayed when the user connects to the database:

```
db2 connect to ldapdb2

SQL1116N A connection to or activation of database <ldapdb2> cannot be made
because of a BACKUP PENDING.
```

The database is in backup pending state until an offline backup is performed. This means the Tivoli Directory Server fails when it connects to the database, and it starts in configuration mode only.

To do a full backup:

```
db2 backup database ldapdb2 to /safeplace
```

If the backup is successful, a message such as the following is displayed:

```
Backup successful.
```

The timestamp for this backup image is: 20040308170601

- Example: Trying to restore a database when ibmslapd is running, the following message is displayed:

  ```
  SQL1035N The database is currently in use.
  ```

- Example: If roll forward must be done following a restore:

  ```
  db2 connect to ldapdb2

  SQL1117N A connection to or activation of database "LDAPDB2" cannot be made
  because of ROLL-FORWARD PENDING. SQLSTATE=57019
  ```

  The database is in roll-forward pending state until a roll-forward command is performed. This means the Tivoli Directory Server fails when it connects to the database, and it starts in configuration mode only.

Optional migration for Tivoli Directory Server V5.2 to V6.0 to support online backup

This section provides example documentation that can be optionally used to migrate Tivoli Directory Server 5.2 databases so that online backup can be used. Because there is no easy way in DB2 to drop a column or reduce the size, the data must be exported out of the old tables and reloaded into the new table definitions. For large databases, this can take a considerable amount of time and resource, which might not be necessary if a customer does not require online backup and point in time recovery support.

These examples are intended to be used by an experienced DB2 database administrator. The commands must be run as the DB2 instance owner (default ldapdb2). We use the example database names ldapdb2 and ldapclog.

In Tivoli Directory Server 5.2 databases, there are four columns defined as BLOB (2G) NOT LOGGED NOT COMPACT as shown in Table E-1.

Table E-1 Tivoli Directory Server v5.2 BLOB (2G) NOT LOGGED NOT COMPACT columns

Database	Table	Column	Used if
ldapdb2	ldap_entry	ENTRYBLOB	entry_size exceeds 24002
ldapdb2	*replchange	DATA_BLOB	data_size exceeds 12002
ldapdb2	*replchange	CONTROL_BLOB	control_size exceeds 12002
ldapclog	ldap_entry	ENTRYBLOB	entry_size exceeds 24004

Note: In Tivoli Directory Server 6.0, the replchange table is replaced by several context-related tables and the new tables are created with 1 GB columns by default.

For directory data that is small enough to not use the blob columns (for example, fit into the varchar column), the varchar columns are logged; however, there is no warning or error issued if that entry is modified and needs to be moved to the not logged blob column. For this reason, online backup should not be used unless all the blob columns are reduced to 1 GB.

Evaluating BLOB columns on Tivoli Directory Server 5.2 and 6.0

blobinfo is an example script, shown in Example E-3, which can be used to analyze Tivoli Directory Server tables and determine information for the columns containing blob definitions for a specific user's directory information. It provides detailed information for the number of entries contained in each type of column. It also provides the current column size definition for the blob columns, the number of entries that are less than or equal to 1 GB, and the number of entries that are greater than 1 GB. For entries greater than 1 GB, it provides a list of entry DNs and entry sizes for those entries, which *cannot* be migrated by the blobmigrate1G script. Those entries must be modified to be less than 1 GB or removed before the blobmigrate1G example script can be successfully run.

Example: E-3 blobinfo script

```
#!/usr/bin/ksh

#############################################################
# blobinfo
#
# This script is provided as an example approach of how to
# analyze affected TDS 5.2 tables and determine counts for
# blobs which are:
#   - entries <= 1G (number of entries which can be migrated)
#   - list of dn's and entrysizes for entries > 1GB
#     (entries which can NOT be migrated)
#
# The blobmigrate1G script can be used as an example approach
# of how to optionally migrate existing 5.2 tables for all
# entries which are <= 1G.
#
# This script should be used by an experienced database
# administrator running as the ldap instance owner.
# This script may need to be modified for different environments.
#
```

```
# This is NOT an officially supported script by IBM or the
# TDS team.
###########################################################
#
# Modify variables below for your environment
###########################################################
# Define Instance, Directory, and Change Log Database
ldapinst=ldapdb2
ldapdb2=LDAPDB2
ldapclog=LDAPCLOG

# Default output directory, if not specified as parameter.
# The ldapinst must have write permission to this directory.
outdir=.

###########################################################
# Usage
numparam=$#
if [ $numparam -ne 1 ]
then
   print "usage: blobinfo <output directory>"
   print "NOTES: If an output directory is NOT specified,"
   print "       the current directory will be used."
else
   outdir=$1
fi

###########################################################
# This script should be run as the ldap instance owner
amildapdb2inst=`whoami | grep $ldapinst`
if [ "X$amildapdb2inst" = "X" ]
then
   print "blobinfo needs to be run as the ldapdb2 instance owner"
   print "which is currently defined to be $ldapinst"
   exit
fi

###########################################################

# Define db2 instance db2profile
. /home/$ldapinst/sqllib/db2profile
###########################################################

# Indicate what logretain is set to
logretainon=`db2 connect to $ldapdb2 >/dev/null;db2 "get db cfg for $ldapdb2" |
grep LOGRETAIN | grep RECOVERY;db2 disconnect $ldapdb2 >/dev/null`
if [ "X$logretainon" != "X" ]
then
   print "LOGRETAIN is currently set to RECOVERY" >> $outdir/blobinfo.rpt
else
   print "LOGRETAIN is NOT set to recovery">> $outdir/blobinfo.rpt
fi

################################################################
```

```
# LDAPDB2 database
# Example - SQL commands (LDAPDB2-LDAP_ENTRY table)
db2 connect to $ldapdb2 >/dev/null
db2 "describe table ldap_entry show detail" >
$outdir/$ldapdb2.ldapentry.beforedescribe
db2 "select count(eid) as total_entries, count(entrydata) as varchar_24004 from
ldap_entry" > $outdir/$ldapdb2.ldapentry.beforecounts
db2 "select count(*) as entryblob_less_1G from ldap_entry where
(entrysize<1073741824 and entrysize>24004)" >>
$outdir/$ldapdb2.ldapentry.beforecounts
db2 "select count(*) as entryblob_greater_1G from ldap_entry where
(entrysize>1073741824)" >> $outdir/$ldapdb2.ldapentry.beforecounts

# Example - Summary Report (blobinfo.rpt)
print "$outdir/blobinfo.rpt" > $outdir/blobinfo.rpt
date >> $outdir/blobinfo.rpt
# Example - Summary Report (LDAPDB2-LDAP_ENTRY table)
print "\n$ldapdb2 Database\n" >> $outdir/blobinfo.rpt
print "Table\t\tTotal#\tTotal#\tBlob\t\tBlob\tTotal#\tTotal#" >>
$outdir/blobinfo.rpt
print "     \t\tEntries\tVarChar\tName\t\tDefn*\tBLOB\tBLOB" >>
$outdir/blobinfo.rpt
print "     \t\t_______\t_______\t___________\t_______\t_<1G___\t_>1G**__" >>
$outdir/blobinfo.rpt
#
lbeforedesc=`cat $outdir/$ldapdb2.ldapentry.beforedescribe | grep ENTRYBLOB | awk
'{print $4}' | awk '{if ($1==2147483647) {print "2G*"} else if ($1==1073741824)
{print "1G"}}'`
totcount=`cat $outdir/$ldapdb2.ldapentry.beforecounts | awk '{if (NR==4) print
$1}'`
totvarchar=`cat $outdir/$ldapdb2.ldapentry.beforecounts | awk '{if (NR==4) print
$2}'`
totblobsmall=`cat $outdir/$ldapdb2.ldapentry.beforecounts | awk '{if (NR==11)
print $1}'`
ltotblobbig=`cat $outdir/$ldapdb2.ldapentry.beforecounts | awk '{if (NR==18) print
$1}'`
print
"LDAP_ENTRY\t"$totcount"\t"$totvarchar"\tENTRYBLOB\t"$lbeforedesc"\t"$totblobsmall
"\t"$ltotblobbig >> $outdir/blobinfo.rpt
# Need to list blobs > 1G if they exist in $ldapdb2 Database
   if [ $ltotblobbig != "0" ]
   then
      print "** Blobs greater than 1G can NOT be supported with online backup">>
$outdir/blobinfo.rpt
      print "** Entries listed below need to be modified or deleted so they" >>
$outdir/blobinfo.rpt
      print "** will fit in a reduced 1G column" >> $outdir/blobinfo.rpt
      db2 "select substr(dn_trunc,1,50) as dn_trunc, entrysize as entrysize from
ldap_entry where (entrysize>=1073741824)" >> $outdir/blobinfo.rpt
   else
      print "\n ** There are NO BLOBs > 1G in the $ldapdb2 Database" >>
$outdir/blobinfo.rpt
fi
db2 disconnect $ldapdb2 >/dev/null
###########################################################
```

```
# The following section is for the Change Log IF it is configured.
# Example - SQL commands
cbeforedesc=notdefined
chglog=`db2 list db directory | grep $ldapclog`
if [ "X$chglog" != "X" ]
then
   db2 "connect to $ldapclog" >/dev/null
   db2 "describe table ldap_entry show detail" >
$outdir/$ldapclog.ldapentry.beforedescribe
   db2 "select count(eid) as total_entries, count(entrydata) as varchar_24004 from
ldap_entry" > $outdir/$ldapclog.ldapentry.beforecounts
   db2 "select count(*) as entryblob_less_1G from ldap_entry where
(entrysize<1073741824 and entrysize>24004)" >>
$outdir/$ldapclog.ldapentry.beforecounts
   db2 "select count(*) as entryblob_greater_1G from ldap_entry where
(entrysize>1073741824)" >> $outdir/$ldapclog.ldapentry.beforecounts
   db2 disconnect $ldapclog >/dev/null

   # Example - Summary Report (LDAPCLOG-LDAP_ENTRY table)
   print "\n$ldapclog Database" >> $outdir/blobinfo.rpt
   print "Table\t\tTotal#\tTotal#\tBlob\t\tBlob\tTotal#\tTotal#" >>
$outdir/blobinfo.rpt
   print "     \t\tEntries\tVarChar\tName\t\tDefn*\tBLOB\tBLOB" >>
$outdir/blobinfo.rpt
   print "     \t\t_______\t_______\t____________\t_______\t_<1G___\t_>1G**__" >>
$outdir/blobinfo.rpt
   cbeforedesc=`cat $outdir/$ldapclog.ldapentry.beforedescribe | grep ENTRYBLOB |
awk '{print $4}' | awk '{if ($1==2147483647) {print "2G*"} else if
($1==1073741824) {print "1G"}}'`
   totcount=`cat $outdir/$ldapclog.ldapentry.beforecounts | awk '{if (NR==4) print
$1}'`
   totvarchar=`cat $outdir/$ldapclog.ldapentry.beforecounts | awk '{if (NR==4)
print $2}'`
   totblobsmall=`cat $outdir/$ldapclog.ldapentry.beforecounts | awk '{if (NR==11)
print $1}'`
   ctotblobbig=`cat $outdir/$ldapclog.ldapentry.beforecounts | awk '{if (NR==18)
print $1}'`
   print
"LDAP_ENTRY\t"$totcount"\t"$totvarchar"\tENTRYBLOB\t"$cbeforedesc"\t"$totblobsmall
"\t"$ctotblobbig >> $outdir/blobinfo.rpt
   # Need to list blobs > 1G if they exist in $ldapclog Database
      if [ $ctotblobbig != "0" ]
      then
         print "** Blobs greater than 1G can NOT be supported with online
backup">> $outdir/blobinfo.rpt
         print "** Entries listed below need to be modified or deleted so they" >>
$outdir/blobinfo.rpt
         print "** will fit in a reduced 1G column" >> $outdir/blobinfo.rpt
         db2 "select substr(dn_trunc,1,50) as dn_trunc, entrysize as entrysize
from ldap_entry where (entrysize>=1073741824)" >> $outdir/blobinfo.rpt
      else
         print "\n ** There are NO BLOBs > 1G in the $ldapclog Database" >>
$outdir/blobinfo.rpt
   fi
fi
```

```
###########################################################
print "\nNotes:" >> $outdir/blobinfo.rpt
# If any column Definition is 2G need to run blobmigrate1G to reduce to 1G
if [ $lbeforedesc = "2G*" ] || [ $cbeforedesc = "2G*" ]
   then
   print " * There is a 1G maximum Blob Definition for Online Backup to be
supported." >> $outdir/blobinfo.rpt
   print " * If you want to have Online Backup supported, you MUST run the
blobmigrate1G" >> $outdir/blobinfo.rpt
   print " * script (see information in the TDS V6.0 Online Backup using" >>
$outdir/blobinfo.rpt
   print " * DB2 Technical White Paper) to migrate them to 1G." >>
$outdir/blobinfo.rpt
fi
print " - LDAP_ENTRY ENTRYBLOB column is used if entry > 24004 " >>
$outdir/blobinfo.rpt
cat $outdir/blobinfo.rpt
```

Example blobinfo.rpt output

The blobinfo script produces an output report named blobinfo.rpt. Let us take a look at some output examples, as shown in Example E-4, Example E-5, and Example E-6.

Example: E-4 Directory database containing no entries with blob greater than 1 GB

```
Thu Jan 20 11:56:47 GMT 2005
LDAPDB2 Database
Table           Total#  Total#  Blob            Blob    Total#  Total#
                Entries VarChar Name            Defn*   BLOB    BLOB
                _______ _______ ___________     _______ _<1G___ _>1G**__
LDAP_ENTRY      1000015 1000015 ENTRYBLOB       2G*     0       0

 ** There are NO BLOBs > 1G in the LDAPDB2 Database
```

Example: E-5 For directory database containing one entry with blob greater than 1 GB

```
Thu Jan 20 12:56:47 GMT 2005
ldapdb2 Database
Table           Total#  Total#  Blob            Blob    Total#  Total#
                Entries VarChar Name            Defn*   BLOB    BLOB
                _______ _______ ___________     _______ _<1G___ _>1G**__
LDAP_ENTRY      1008    1006    ENTRYBLOB       2G*     1       1

** Blobs greater than 1G can NOT be supported with online backup
** Entries listed below need to be modified or deleted so they
** will fit in a reduced 1G column
DN_TRUNC                                           ENTRYSIZE
-------------------------------------------------- -----------
CN=BIGENTRY, O=BIGENTRIES                           1073741829

  1 record(s) selected.
```

```
Notes:
 * There is a 1G maximum Blob Definition for Online Backup to be supported.
 * If you want to have Online Backup supported, you MUST run the blobmigrate1G
 * script (see information in the Tivoli Directory Server V6.0 Online Backup using
 * DB2 Technical White Paper) to migrate them to 1G.
 - LDAP_ENTRY ENTRYBLOB column is used if entry > 24004
```

Example: E-6 For directory and change log database containing no entries with blob greater than 1 GB

```
Sun Jun 27 20:55:03 CDT 2004
LDAPDB2 Database
Table           Total#  Total#  Blob            Blob    Total#  Total#
                Entries VarChar Name            Defn*   BLOB    BLOB
                _______ _______ ___________     _______ _<1G___ _>1G**__
LDAP_ENTRY      1000015 1000015 ENTRYBLOB       2G*     0       0

 ** There are NO BLOBs > 1G in the LDAPDB2 Database

LDAPCLOG Database
Table           Total#  Total#  Blob            Blob    Total#  Total#
                Entries VarChar Name            Defn*   BLOB    BLOB
                _______ _______ ___________     _______ _<1G___ _>1G**__
LDAP_ENTRY      2       2       ENTRYBLOB       2G*     0       0

 ** There are NO BLOBs > 1G in the LDAPCLOG Database
```

Example blobinfo.rpt notes

In the blobinfo.rpt output, note that:

- In Example E-4, the blobmigrate1G (described in the following section) script can be run because there are *no* entries with blobs greater than 1 GB.
- In Example E-5, the entry with a blob greater than 1 GB must be deleted before the blobmigrate1G script can be run.
- In Example E-6, the blobmigrate1G script can be run because there are *no* entries with blobs greater than 1 GB.

blobmigrate1G script

blobmigrate1G is an example script that can be used to *optionally migrate* migrated 5.2 tables for all entries that are less than or equal to 1 GB in size. This migration is being done *optionally* because there is no easy way to drop a column or reduce the size, therefore the data must be exported out of the old tables and reloaded into the new tables. A full backup of the database is strongly suggested as an error recovery strategy.

Large file support on AIX for JFS and/or use JFS2

The standard file system on some older versions of AIX has a 2 GB file size limit, regardless of the **ulimit** setting. One way to enable files larger than the 2 GB limit is to create the file system with the Large File Enabled option. This option can be found through the Add a Journaled File System option of the smit menu.

Newer versions of AIX support Enhanced Journaled File Systems (JFS2), which by default support files larger than 2 GB. This option should be chosen when the file system is created.

Refer to the AIX documentation for additional information and file system options.

Example E-7 shows an example of our blobmigrate1G script.

Example: E-7 blobmigrate1G script

```
#!/usr/bin/ksh
############################################################
# blobmigrate1G
#
# This script is provided as an example approach of how to
# "optionally" migrate existing 5.2 tables for all entries
# which are <= 1GB.
#
# This script must be run AFTER V6.0 migration has been
# successfully completed.
#
# This migration is being done optionally because there is
# no easy way to drop a column or reduce the size, so the
# data will need to be exported out of the old tables and
# reloaded into the new tables.
#
# A full backup of the database(s) will be made as an error
# recovery strategy
#
# This script should be used by an experienced database
# administrator running as the ldapdb2 instance owner.
# It may need to be modified for different environments.
#
# This script is NOT an officially supported script by IBM
# or the TDS team.
############################################################

# Modify variables below for your environment
############################################################
# Define Instance, Directory, and Change Log Database
ldapinst=ldapdb2
ldapdb2=LDAPDB2
ldapclog=LDAPCLOG
#
# Define Config and schema file location - Make backup copy?
# V6.0
LDAPconfig=/home/$ldapinst/idsslapd-ldapdb2/etc/ibmslapd.conf
LDAPschemaloc=/home/ldapdb2/idsslapd-ldapdb2/etc
# V5.2
#LDAPconfig=/etc/ibmslapd.conf
#LDAPschemaloc=/etc/ldapschema
#
# Directory to use to store backups and working space
# The ldapinst must have write permission to this directory
outdir=$1
############################################################

numparam=$#
if [ $numparam -ne 1 ]
then
   print "usage: blobmigrate1G <output directory>"
   print "NOTES: 1) Script must be run as ldap instance owner"
   print "       2) TDS V6.0 Migration must by DONE"
   print "       3) ibmslapd must NOT be running"
   print "       4) logretain should be turned OFF"
   print "       5) blobs > 1G can NOT be migrated"
   print "       6) databases ldapdb2 and ldapclog must NOT be in use"
   print ""
   print " The summary report is <output directory> blobmigrate1G.rpt"
   exit
fi
############################################################
. /home/$ldapinst/sqllib/db2profile
############################################################
#
#1 - This script must be run as the ldap instance owner
amildapdb2inst=`whoami | grep $ldapinst`
if [ "X$amildapdb2inst" = "X" ]
then
   print "blobmigrate1G needs to be run as the ldapdb2 instance owner"
   print "which is currently defined to be $ldapinst"
   exit
fi

############################################################
#2 - Need to exit if TDS V6.0 Migration is NOT Done
# itdsrdbmhistory table does not exist prior to TDS V6.0
#v6migdone=`db2 connect to $ldapdb2 >/dev/null;db2 "select count(*) from
itdsrdbmhistory where (release='6.0' and feature='MIGRATION_DONE')" | awk '{if (NR
== 4) print $1}';db2 disconnect $ldapdb2 >/dev/null`
#if [ $v6migdone != "0" ]
#then
#   print "TDS V6.0 Migration must be DONE prior to doing blobmigrate1G."
#   print "Make sure the TDS server starts successfully prior to doing
blobmigrate."
#   exit
#fi

############################################################
#3 - Need to exit if ibmslapd is running
ibmslapdup=`ps -ef | grep -i ibmslapd | grep -v grep`
if [ "X$ibmslapdup" != "X" ]
then
   echo "ibmslapd must NOT be running for blobmigrate1G to work"
   exit
fi

############################################################
#4 - Need to exit if logretain is ON for either database
logretainon=`db2 connect to $ldapdb2 >/dev/null;db2 "get db cfg for $ldapdb2" |
grep LOGRETAIN | egrep "ON|RECOVERY";db2 disconnect $ldapdb2 >/dev/null`
clogretainon=`db2 connect to $ldapclog >/dev/null;db2 "get db cfg for $ldapclog" |
grep LOGRETAIN | egrep "ON|RECOVERY";db2 disconnect $ldapclog >/dev/null`
if [ "X$logretainon" != "X" ] || [ "X$clogretainon" != "X" ]
then
   print "LOGRETAIN should NOT be ON when doing blobmigrate1G"
   print "Need to do: db2 update db cfg for $ldapdb2 using logretain off "
   exit
fi

###########################################################
#5 - Need to exit if blobs > 1G exist
numbigblobs=`db2 connect to $ldapdb2 >/dev/null;db2 "select count(*) as
entryblob_greater_1G from ldap_entry where (entrysize>1073741824)" | awk '{if (NR
== 4) print $1}';db2 disconnect $ldapdb2 >/dev/null`
if [ $numbigblobs != "0" ]
then
   print "blobmigrate1G CAN NOT migrate blobs greater than 1G"
   print "Use blobinfo to identify the entries which are > 1G"
   print "Modify or delete them so they will fit in a reduced 1G column"
   exit
fi

###########################################################
#6 - If ldapdb2 or ldapclog are in use - force will disconnect
db2 force applications all >/dev/null

###########################################################
# May want to Backup Config and Schema Files
# Need to have permission to copy files
# cp $LDAPconfig $outdir
# cp -r $LDAPschemaloc/* $outdir
###########################################################

# blobmigrate1G output files in $outdir
#
# Summary Report:
#     blobmigrate1G.rpt
#
# Servicability and Debug Files:
#
#   ldapdb2
#     ldapdb2.before.describe & ldapdb2.after.describe
#     ldapdb2.before.counts   & ldapdb2.after.counts
#     newLDAPentry - exported data for new ldapdb2 ldap_entry table
#     LDAPDB2.load.msg - db2 load messages
#     LDAPDB2.reorgchk.done - reorgchk after successful completion
#
#   ldapclog
#     ldapclog.before.describe & ldapclog.after.describe
#     ldapclog.before.counts   & ldapclog.after.counts
#     newldapclogLDAPentry - exported data for new ldapclog ldap_entry table
#     LDAPCLOG.load.msg - db2 load messages
#     LDAPCLOG.reorgchk.done - reorgchk after successful completion
#
#   overall
#     blobmigrate1G.sql - sql command output
#     blobmigrate1G.diff - diff command output
#

###########################################################################
```

```
# Backup ldapdb2 database
print "********************************" > $outdir/blobmigrate1G.rpt
print "$outdir/blobmigrate1G.rpt" >> $outdir/blobmigrate1G.rpt
print "\nBegin backup database $ldapdb2 to $outdir at: " >>
$outdir/blobmigrate1G.rpt
date >> $outdir/blobmigrate1G.rpt
db2 backup database $ldapdb2 to $outdir >> $outdir/blobmigrate1G.rpt
print "End backup database $ldapdb2 to $outdir at: " >> $outdir/blobmigrate1G.rpt
date >> $outdir/blobmigrate1G.rpt

# Collect ldapdb2 BEFORE info
db2 connect to $ldapdb2  > $outdir/blobmigrate1G.sql
db2 "describe table ldap_entry show detail" >
$outdir/$ldapdb2.ldapentry.beforedescribe
db2 "select count(eid) as total_entries, count(entrydata) as varchar_24004 from
ldap_entry" > $outdir/$ldapdb2.ldapentry.beforecounts
db2 "select count(*) as entryblob_less_1G from ldap_entry where
(entrysize<1073741824 and entrysize>24004)" >>
$outdir/$ldapdb2.ldapentry.beforecounts
db2 "select count(*) as entryblob_greater_1G from ldap_entry where
(entrysize>1073741824)" >> $outdir/$ldapdb2.ldapentry.beforecounts
# Export ldapdb2 data for ldap_entry table
db2 "export to $outdir/newLDAPentry of del lobs to $outdir/ modified by lobsinfile
messages $outdir/$ldapdb2.export.msg select eid, peid, dn_trunc, dn, creator,
modifier, modify_timestamp, create_timestamp, entrydata, entryblob, entrysize from
ldap_entry" >> $outdir/blobmigrate1G.sql
# Rename table and indexes
db2 "rename table ldap_entry to ldap_entry_old" >> $outdir/blobmigrate1G.sql
db2 "rename index ldap_entry_peid to ldap_entry_peido" >>
$outdir/blobmigrate1G.sql
db2 "rename index ldap_entry_peid2 to ldap_entry_peid2o" >>
$outdir/blobmigrate1G.sql
db2 "rename index ldap_entry_trunc to ldap_entry_trunco" >>
$outdir/blobmigrate1G.sql
# Create new table and indexes
db2 "create table ldap_entry(eid integer not null,peid integer,dn_trunc
varchar(240),dn varchar(1000),creator varchar(1000),modifier
varchar(1000),modify_timestamp timestamp,create_timestamp timestamp,entrydata
varchar(24004),entryblob blob(1g) logged not compact,entrysize integer) in
ldapspace" >> $outdir/blobmigrate1G.sql
db2 "create index ldap_entry_peid on $ldapdb2.ldap_entry (eid asc, peid asc)"  >>
$outdir/blobmigrate1G.sql
db2 "create index ldap_entry_peid2 on $ldapdb2.ldap_entry (peid asc)" >>
$outdir/blobmigrate1G.sql
db2 "create index ldap_entry_trunc on $ldapdb2.ldap_entry (dn_trunc asc)" >>
$outdir/blobmigrate1G.sql
db2 "alter table ldap_entry add primary key (eid)" >> $outdir/blobmigrate1G.sql
# Load Data from exported file
db2 "load from $outdir/newLDAPentry of del lobs from $outdir/ modified by
lobsinfile delprioritychar fastparse savecount 100000 warningcount 1 messages
$outdir/$ldapdb2.load.msg insert into $ldapdb2.ldap_entry(eid, peid, dn_trunc, dn,
creator, modifier, modify_timestamp, create_timestamp, entrydata, entryblob,
entrysize)" >> $outdir/blobmigrate1G.sql
# Collect ldapdb2 AFTER info
```

```
db2 "describe table ldap_entry show detail" >
$outdir/$ldapdb2.ldapentry.afterdescribe
db2 "select count(eid) as total_entries, count(entrydata) as varchar_24004 from
ldap_entry" > $outdir/$ldapdb2.ldapentry.aftercounts
db2 "select count(*) as entryblob_less_1G from ldap_entry where
(entrysize<1073741824 and entrysize>24004)" >>
$outdir/$ldapdb2.ldapentry.aftercounts
db2 "select count(*) as entryblob_greater_1G from ldap_entry where
(entrysize>1073741824)" >> $outdir/$ldapdb2.ldapentry.aftercounts
# Summary
print "\nStatistics for blobmigrate1G from OLD to NEW:" >>
$outdir/blobmigrate1G.rpt
print "\n$ldapdb2 Database\n" >> $outdir/blobmigrate1G.rpt
print "State\tTable\t\tTotal#\tTotal#\tBlob\tTotal#\tTotal#" >>
$outdir/blobmigrate1G.rpt
print "     \t     \t\tEntries\tVarChar\tDefn\tBLOB<1G\tBLOB>1G" >>
$outdir/blobmigrate1G.rpt
print "     \t     \t\t_______\t_______\t____\t_______\t_______" >>
$outdir/blobmigrate1G.rpt
#
lbeforedesc=`cat $outdir/$ldapdb2.ldapentry.beforedescribe | grep ENTRYBLOB | awk
'{print $4}' | awk '{if ($1==2147483647) {print "2G"} else if ($1==1073741824)
{print "1G"}}'`
totcount=`cat $outdir/$ldapdb2.ldapentry.beforecounts | awk '{if (NR==4) print
$1}'`
totvarchar=`cat $outdir/$ldapdb2.ldapentry.beforecounts | awk '{if (NR==4) print
$2}'`
totblobsmall=`cat $outdir/$ldapdb2.ldapentry.beforecounts | awk '{if (NR==11)
print $1}'`
totblobbig=`cat $outdir/$ldapdb2.ldapentry.beforecounts | awk '{if (NR==18) print
$1}'`
print
"OLD\tLDAP_ENTRY\t"$totcount"\t"$totvarchar"\t"$lbeforedesc"\t"$totblobsmall"\t"$t
otblobbig >> $outdir/blobmigrate1G.rpt
lafterdesc=`cat $outdir/$ldapdb2.ldapentry.afterdescribe | grep ENTRYBLOB | awk
'{print $4}' | awk '{if ($1==2147483647) {print "2G"} else if ($1==1073741824)
{print "1G"}}'`
totcount=`cat $outdir/$ldapdb2.ldapentry.aftercounts | awk '{if (NR==4) print
$1}'`
totvarchar=`cat $outdir/$ldapdb2.ldapentry.aftercounts | awk '{if (NR==4) print
$2}'`
totblobsmall=`cat $outdir/$ldapdb2.ldapentry.aftercounts | awk '{if (NR==11) print
$1}'`
totblobbig=`cat $outdir/$ldapdb2.ldapentry.aftercounts | awk '{if (NR==18) print
$1}'`
print
"NEW\tLDAP_ENTRY\t"$totcount"\t"$totvarchar"\t"$lafterdesc"\t"$totblobsmall"\t"$to
tblobbig >> $outdir/blobmigrate1G.rpt
db2 disconnect $ldapdb2 >> $outdir/blobmigrate1G.sql
##########################################################
# The following section is for the Change Log IF it is configured.
chglog=`db2 list db directory | grep $ldapclog`
if [ "X$chglog" != "X" ]
then
   # Backup ldapclog database
   print "\nBegin backup database $ldapclog to $outdir at " >>
$outdir/blobmigrate1G.rpt
   date >> $outdir/blobmigrate1G.rpt
   db2 backup database $ldapclog to $outdir >> $outdir/blobmigrate1G.rpt
   print "End backup database $ldapclog to $outdir at " >>
$outdir/blobmigrate1G.rpt
   date >> $outdir/blobmigrate1G.rpt

   # Collect ldapclog BEFORE info
   db2 connect to $ldapclog >> $outdir/blobmigrate1G.sql
   db2 "describe table ldap_entry show detail" >
$outdir/$ldapclog.ldapentry.beforedescribe
   db2 "select count(eid) as total_entries, count(entrydata) as varchar_24004 from
ldap_entry" > $outdir/$ldapclog.ldapentry.beforecounts
   db2 "select count(*) as entryblob_less_1G from ldap_entry where
(entrysize<1073741824 and entrysize>24004)" >>
$outdir/$ldapclog.ldapentry.beforecounts
   db2 "select count(*) as entryblob_greater_1G from ldap_entry where
(entrysize>1073741824)" >> $outdir/$ldapclog.ldapentry.beforecounts
   # Export ldapclog data for ldap_entry table
   db2 "export to $outdir/newldapclogLDAPentry of del lobs to $outdir/ modified by
lobsinfile messages $outdir/$ldapclog.export.msg select eid, peid, dn_trunc, dn,
creator, modifier, modify_timestamp, create_timestamp, entrydata, entryblob,
entrysize from ldap_entry" >> $outdir/blobmigrate1G.sql
  # Rename tables and indexes
   db2 "rename table ldap_entry to ldap_entry_old" >> $outdir/blobmigrate1G.sql
   db2 "rename index ldap_entry_peid to ldap_entry_peido" >>
$outdir/blobmigrate1G.sql
   db2 "rename index ldap_entry_peid2 to ldap_entry_peid2o" >>
$outdir/blobmigrate1G.sql
   db2 "rename index ldap_entry_trunc to ldap_entry_trunco" >>
$outdir/blobmigrate1G.sql
   # Create new table and indexes
   db2 "create table ldap_entry(eid integer not null,peid integer,dn_trunc
varchar(240),dn varchar(1000),creator varchar(1000),modifier
varchar(1000),modify_timestamp timestamp,create_timestamp timestamp,entrydata
varchar(24004),entryblob blob(1g) logged not compact,entrysize integer) in
ldapspace" >> $outdir/blobmigrate1G.sql
   db2 "create index ldap_entry_peid on $ldapdb2.ldap_entry (eid asc, peid asc)"
>> $outdir/blobmigrate1G.sql
   db2 "create index ldap_entry_peid2 on $ldapdb2.ldap_entry (peid asc)" >>
$outdir/blobmigrate1G.sql
   db2 "create index ldap_entry_trunc on $ldapdb2.ldap_entry (dn_trunc asc)" >>
$outdir/blobmigrate1G.sql
   db2 "alter table ldap_entry add primary key (eid)" >> $outdir/blobmigrate1G.sql
   # Load data from exported file
   db2 "load from $outdir/newldapclogLDAPentry of del lobs from $outdir/ modified
by lobsinfile delprioritychar fastparse savecount 100000 warningcount 1 messages
$outdir/$ldapclog.load.msg insert into $ldapdb2.ldap_entry(eid, peid, dn_trunc,
dn, creator, modifier, modify_timestamp, create_timestamp, entrydata, entryblob,
entrysize)" >> $outdir/blobmigrate1G.sql
   db2 "describe table ldap_entry show detail" >
$outdir/$ldapclog.ldapentry.afterdescribe
   # Collect ldapclog AFTER info
   db2 "select count(eid) as total_entries, count(entrydata) as varchar_24004 from
ldap_entry" > $outdir/$ldapclog.ldapentry.aftercounts
   db2 "select count(*) as entryblob_less_1G from ldap_entry where
(entrysize<1073741824 and entrysize>24004)" >>
$outdir/$ldapclog.ldapentry.aftercounts
   db2 "select count(*) as entryblob_greater_1G from ldap_entry where
(entrysize>1073741824)" >> $outdir/$ldapclog.ldapentry.aftercounts
   db2 disconnect $ldapclog >> $outdir/blobmigrate1G.sql
   # Summary
   print "\n$ldapclog Database" >> $outdir/blobmigrate1G.rpt
   print "State\tTable\t\tTotal#\tTotal#\tBlob\tTotal#\tTotal#" >>
$outdir/blobmigrate1G.rpt
   print "     \t     \t\tEntries\tVarChar\tDefn\tBLOB<1G\tBLOB>1G" >>
$outdir/blobmigrate1G.rpt
   print "     \t     \t\t_______\t_______\t____\t_______\t_______" >>
$outdir/blobmigrate1G.rpt
   cbeforedesc=`cat $outdir/$ldapclog.ldapentry.beforedescribe | grep ENTRYBLOB |
awk '{print $4}' | awk '{if ($1==2147483647) {print "2G"} else if ($1==1073741824)
{print "1G"}}'`
   totcount=`cat $outdir/$ldapclog.ldapentry.beforecounts | awk '{if (NR==4) print
$1}'`
   totvarchar=`cat $outdir/$ldapclog.ldapentry.beforecounts | awk '{if (NR==4)
print $2}'`
   totblobsmall=`cat $outdir/$ldapclog.ldapentry.beforecounts | awk '{if (NR==11)
print $1}'`
   totblobbig=`cat $outdir/$ldapclog.ldapentry.beforecounts | awk '{if (NR==18)
print $1}'`
   print
"OLD\tLDAP_ENTRY\t"$totcount"\t"$totvarchar"\t"$cbeforedesc"\t"$totblobsmall"\t"$t
otblobbig >> $outdir/blobmigrate1G.rpt
   cafterdesc=`cat $outdir/$ldapclog.ldapentry.afterdescribe | grep ENTRYBLOB |
awk '{print $4}' | awk '{if ($1==2147483647) {print "2G"} else if ($1==1073741824)
{print "1G"}}'`
   totcount=`cat $outdir/$ldapclog.ldapentry.aftercounts | awk '{if (NR==4) print
$1}'`
   totvarchar=`cat $outdir/$ldapclog.ldapentry.aftercounts | awk '{if (NR==4)
print $2}'`
   totblobsmall=`cat $outdir/$ldapclog.ldapentry.aftercounts | awk '{if (NR==11)
print $1}'`
   totblobbig=`cat $outdir/$ldapclog.ldapentry.aftercounts | awk '{if (NR==18)
print $1}'`
   print
"NEW\tLDAP_ENTRY\t"$totcount"\t"$totvarchar"\t"$cafterdesc"\t"$totblobsmall"\t"$to
tblobbig >> $outdir/blobmigrate1G.rpt
fi
print "\nNote:" >> $outdir/blobmigrate1G.rpt
print "LDAP_ENTRY has 1 BLOB column which is used if entry > 24004\t" >>
$outdir/blobmigrate1G.rpt
########################################################################
# Determine if successful by looking at difference between output files.
# Only differences should be 2G vs 1G column size from describe files.
# Both ldapdb2 and ldapclog (if configured) entry counts must be the same
# before and after to be successful.  See blobmigrate1G.diff for differences.
# If NOT successful, "db2 restore db $ldapdb2 from $outdir" and/or
#                    "db2 restore db $ldapclog from $outdir"
```

```
print "\n\n*****************************"  >> $outdir/blobmigrate1G.rpt
print "*** Overall results are: ***"  >> $outdir/blobmigrate1G.rpt
print "*****************************"  >> $outdir/blobmigrate1G.rpt
# ldapdb2
print "Diff between $ldapdb2.ldapentry.beforedescribe and
$ldapdb2.ldapentry.afterdescribe\n" > $outdir/blobmigrate1G.diff
diff $outdir/$ldapdb2.ldapentry.beforedescribe
$outdir/$ldapdb2.ldapentry.afterdescribe >> $outdir/blobmigrate1G.diff
print "$ldapdb2 LDAP_ENTRY - ENTRYBLOB changed from $lbeforedesc to $lafterdesc"
>> $outdir/blobmigrate1G.rpt
print "Diff between $ldapdb2.ldapentry.beforecounts and
$ldapdb2.ldapentry.aftercounts\n"  >> $outdir/blobmigrate1G.diff
diffentry=`diff $outdir/$ldapdb2.ldapentry.beforecounts
$outdir/$ldapdb2.ldapentry.aftercounts`
if [ "X$diffentry" != "X" ]
then
   print "**Differences found for $ldapdb2 ldapentry counts" >>
$outdir/blobmigrate1G.rpt
   print "**blobmigrate1G was NOT successful\n\n" >> $outdir/blobmigrate1G.rpt
   exit
else
   print "$ldapdb2 LDAP_ENTRY - All Entry Counts OK \n" >>
$outdir/blobmigrate1G.rpt
fi
diffclog=""
if [ "X$chglog" != "X" ]
then
   print "Diff between $ldapclog.ldapentry.beforedescribe and
$ldapclog.ldapentry.afterdescribe" >> $outdir/blobmigrate1G.diff
   diff $outdir/$ldapclog.ldapentry.beforedescribe
$outdir/$ldapclog.ldapentry.afterdescribe >> $outdir/blobmigrate1G.diff
   print "$ldapclog LDAP_ENTRY - ENTRYBLOB changed from $lbeforedesc to
$lafterdesc" >> $outdir/blobmigrate1G.rpt
   print "Diff between $ldapclog.ldapentry.beforecounts and
$ldapclog.ldapentry.aftercounts"  >> $outdir/blobmigrate1G.diff
   diffclog=`diff $outdir/$ldapclog.ldapentry.beforecounts
$outdir/$ldapclog.ldapentry.aftercounts`
   if [ "X$diffclog" != "X" ]
   then
   print "**Differences found for $ldapclog ldapentry counts" >>
$outdir/blobmigrate1G.rpt
   print "**blobmigrate1G was NOT successful\n\n" >> $outdir/blobmigrate1G.rpt
   exit
else
   print "$ldapclog LDAP_ENTRY - All Entry Counts OK \n" >>
$outdir/blobmigrate1G.rpt
   fi
else
   print "$ldapclog is NOT configured" >> $outdir/blobmigrate1G.rpt
fi
# IF everything goes successfully, need to update itdsrdbmhistory table
# and drop renamed old tables
if [ "X$diffentry" = "X" ] && [ "X$diffclog" = "X" ]
then
   db2 connect to $ldapdb2 >> $outdir/blobmigrate1G.sql
   # The following table is new for TDS V6.0
   db2 "insert into itdsrdbmhistory values ('6.0','BLOBMIGRATE1G_DONE','')" >>
$outdir/blobmigrate1G.sql
   db2 drop table ldap_entry_old >> $outdir/blobmigrate1G.sql
   db2 reorgchk update statistics on table all > $outdir/$ldapdb2.reorgchk.done
   db2 disconnect $ldapdb2
   if [ "X$chglog" != "X" ]
   then
      db2 connect to $ldapclog >> $outdir/blobmigrate1G.sql
      db2 drop table ldap_entry_old >> $outdir/blobmigrate1G.sql
      db2 reorgchk update statistics on table all > $outdir/$ldapdb2.reorgchk.done
      db2 disconnect $ldapclog >> $outdir/blobmigrate1G.sql
   fi
   print "\n*** blobmigrate1G was SUCCESSFUL! ***" >> $outdir/blobmigrate1G.rpt
   print "*** After verifying everything works correctly with TDS" >>
$outdir/blobmigrate1G.rpt
   print "*** you may remove the directory $outdir\n" >> $outdir/blobmigrate1G.rpt
else
   print "\n*** blobmigrate1G was NOT successful. ***\n" >>
$outdir/blobmigrate1G.rpt
   print "*** Everything has been saved in directory $outdir\n" >>
$outdir/blobmigrate1G.rpt
   print "*** including database backups and informational files.\n" >>
$outdir/blobmigrate1G.rpt
fi
cat $outdir/blobmigrate1G.rpt
```

Example E-8 shows an example blobmigrate1G.rpt output.

Example: E-8 Directory database containing no entries with blob greater than 1 GB

```
./blobmigrate1G /outdir
Begin backup database ldapdb2 to /outdir at:
Thu Jan 20 13:11:11 GMT 2005
Backup successful. The timestamp for this backup image is: 20050120131111
End backup database ldapdb2 to /outdir at:
Thu Jan 20 13:11:48 GMT 2005
Statistics for blobmigrate1G from OLD to NEW:
ldapdb2 Database
State   Table           Total#  Total#  Blob    Total#  Total#
                        Entries VarChar Defn    BLOB<1G BLOB>1G
                        _______ _______ ____    _______ _______
OLD     LDAP_ENTRY      1010    1009    2G      1       0
NEW     LDAP_ENTRY      1010    1009    1G      1       0
Note:
LDAP_ENTRY has 1 BLOB column which is used if entry > 24004
****************************
*** Overall results are: ***
****************************
ldapdb2 LDAP_ENTRY - ENTRYBLOB changed from 2G to 1G
ldapdb2 LDAP_ENTRY - All Entry Counts OK
LDAPCLOG is NOT configured
*** blobmigrate1G was SUCCESSFUL! ***
*** After verifying everything works correctly with TDS
*** you may remove the directory /outdir
```

F

Checklist

In this appendix, we provide you with several checklists covering maintenance, monitoring, and other Lightweight Directory Access Protocol (LDAP) utilities.

Maintenance checklist

We recommend the following Tivoli Directory Server maintenance steps:

- Directory server backup daily.

  Refer to Appendix E, “Online backup of Tivoli Directory Server” on page 143 for details.

  - Database backup
  - Tivoli Directory Server important files backup
  - Backup of the current day's changes files
- Rotate and examine Tivoli Directory Server log files. You can write a script to do the file name rotation.

  - audit.log
  - ibmslapd.log
  - db2diag.log file

  You might have to check other log files if there is any error, but might not have to rotate the other log files.

  For details in problem determination, refer to *IBM Tivoli Directory Server Problem Determination Guide Version 6.0*, SC32-1679.
- Check if the hard disk has enough space for log files and database to grow.
- Performance check every month. Tune the directory server if needed.
- Monitor Tivoli Directory Server status using either Web administration tool or command line utilities.
- Prepare directory server failover procedure.
- Prepare directory server backup and restore procedure.
- Check the latest Tivoli Directory Server fix pack or upgrade.

IBM Tivoli Directory Server support tool

The support tool collects relevant data such as logs, directory listings, schema files, and core files about the directory server. The support tool then packages the information into a compressed file archive that you can send to IBM Software Support for help in diagnosing your problem. The *IBM Tivoli Directory Server Problem Determination Guide Version 6.0*, SC32-1679, describes the information collected by the support tool, and contains instructions for generating the idssupport file.

Monitoring Tivoli Directory Server checklist

Monitoring Tivoli Directory Server for problems can be broken down into several different categories:

- Monitoring for outages or performance degradations:

  Tivoli Directory Server consist of two primary processes that must be active and consuming resources for Tivoli Directory Server to be healthy:

  - The LDAP server process (a unique name reflecting the instance name of Tivoli Directory Server)
  - UDB/DB2

- Monitoring for performance:
  - auditmonit.jar
  - ldapsearch utility
- Monitor Tivoli Directory Server status using either Web administration tool or command line utilities such as ldapsearch or Tivoli Directory Integrator monitoring utility.
- Analyze the Tivoli Directory Server log files to check the status of directory server periodically.
- Monitoring Tivoli Directory Server performance.
- Optional:

  You can also set up Tivoli Directory Integrator AssemblyLines to monitor Tivoli Directory Server status and send an automated e-mail to the system administrator when Tivoli Directory Server fails, but this requires additional effort to build such function in Tivoli Directory Integrator.
- Optional:

  You can also use other IBM Tivoli professional monitoring products; this requires additional effort to set up such service.

Like any application, the ability of an administrator to understand what the current state of the application at any given time is critical.

The monitoring of the directory is important from the following perspectives:

- Security issues

  To track unauthorized access and take corrective measures. Let us take some example where we want to prevent unauthorized access to the directory. Monitoring the server helps to overcome issues such as these and consequently secure our server.
- Performance issues

  To obtain reasons for slow or poor performance.

  Performance of the directory server might be low owing to many reasons:

  - The DB2 buffer pools might not be tuned as per the directory requirements, though there are resources available.
  - The Directory Server caches might not be tuned to the optimum.
  - There might be some important indexes missing.
  - There might be a database reorganization required and many more.

From the above list, some of the problems might be caught or prevented using the monitoring tools. Monitoring tools help us in deriving the optimal values for a set of directory parameters after statistical analysis of the existing workload and resources.

- Throughput measurement: To derive statistics such as how many searches have been completed in a given time frame, how many additions have been completed, how many binds have occurred to the server, how many operations have been completed. Keeping track of such figures helps us to calculate the throughput of the server. This throughput might be quite influential for the dependent products that are going to use the directory server.

The Tivoli Directory Server is provided with a set of tools that can be used to monitor the directory server against any anomalies. The Tivoli Directory Server monitoring can be accomplished in many ways as listed below:

- Searching against the base cn=monitor
- Searching through the change log database
- Analyzing log files

Let us now begin to look into some of the details of monitoring Tivoli Directory Server.

Monitoring for outages and performance degradations

Tivoli Directory Server consist of two primary processes that must be active and consuming resources for Tivoli Directory Server to be healthy:

- The LDAP server process (a unique name reflecting the instance name of Tivoli Directory Server)
- UDB/DB2

Monitoring the LDAP server process

Monitoring the LDAP server process can be broken down into five distinct areas:

- The LDAP server process (instance name is chosen at instance creation) must be active and in memory. If this process dies, the monitoring tool chosen should attempt at least one iteration of an automated restart and log the event.
- Monitoring log files:

  Tivoli Directory Server's critical log files must be monitored for operational issues.

  - <Instance Name Log>

    This log file must be monitored for:

    - Correct startup (for example, not in configuration mode)
    - Replication queues active and remote login successful
    - UDB/DB2 started and accepting connections

  - Audit.log:

    This log file (instance specific) should be configured (errors only) and monitored for error conditions such as:

    - Excessive failures of root account binds.
    - Abandoned connections (usually due to client, network, or Tivoli Directory Server not responding in time).

- Query of Tivoli Directory Server's cn=monitor object:

  The cn=monitor object should be read on a regular basis (for example, once a minute) to determine possible error conditions such as:

  - Emergency thread active
  - Depth of work queue
  - Opscompleted is incrementing (when client applications are not in maintenance/failure mode)

- Synthetic transactions:

  Synthetic transactions (representative queries of a random nature) should be applied to the directory to ensure that it is meeting established service-level agreements (SLAs).

Note: These queries should be as random as possible to ensure that cached results are not being returned.

- Monitoring file space:

  Critical files spaces (used to store log, configuration, and databases) must be monitored to ensure that excessive growth is captured before a file system full condition causes an outage.

Monitoring the UDB/DB2 process

Monitoring the UDB/DB2 process can be broken down into three distinct areas:

- The UDB/DB2 server process must be active and in memory. If this process dies, it should be automatically restarted as it is normally configured in the inittab. The monitoring tool chosen should at least log the event, and send an alert if the process is not restarted within 30 seconds.
- Monitoring log files:

  DB2's critical log files must be monitored for operational issues.

  - Idsldap.nfy:

    A shortened version of the db2diag.log file that only contains errors.

    - Correct startup
    - Error conditions such as out of space and so on
  - Db2diag.log:

    This log file should be monitored to ensure that standard processes (reorg or tuning operations are completing successfully)
- Monitoring file space:

  Critical files spaces (used to store log, configuration and databases) must be monitored to ensure that excessive growth is captured before a file system full condition causes an outage.

Monitoring for performance and SLA conformance

The best method (and only recommended method for large directories) to determine the average performance of operations is through the use of synthetic transactions. Using a pseudo-random set of LDAP operations (binds, unbinds, adds, deletes, modifies, and searches) allows you to capture and log the performance of the directory over a period of time. The only critical factors in the development of these synthetic transactions is:

- They should be run on the same server as the directory (or proxy) to ensure that network latency is not an issue.
- They should be random in nature to ensure that caches do not skew the data.
- They are of a lightweight nature (both the operations and the initiating application) such that they do not impact the production performance of the directory.

Monitoring Tivoli Directory Server status

Monitoring of the Tivoli Directory Server status can be done by using either Web administration tool or command line utilities such as the ldapsearch utility.

Operating system command line to monitor Tivoli Directory Server

Command line utility to view information about the running ibmslapd process:

```
ps -eaf | grep -i ibmslapd
```

The above command helps us to get to know two things:

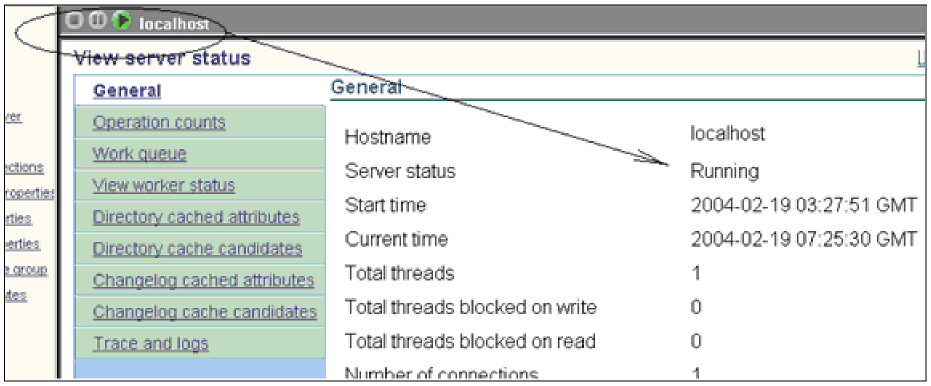
- Firstly, if ibmslapd is still running.
- Secondly, how much the process size has grown till date and if it is within permissible limits or going to click the limits soon.

Web administration tool

You can use the Web administration tool to:

- View server status

  You have to log in to the Web administration tool in order to view the server status as shown in Figure F-1.



localhost
View server status
General
Operation counts
Work queue
View worker status
Directory cached attributes
Directory cache candidates
Changelog cached attributes
Changelog cache candidates
Trace and logs

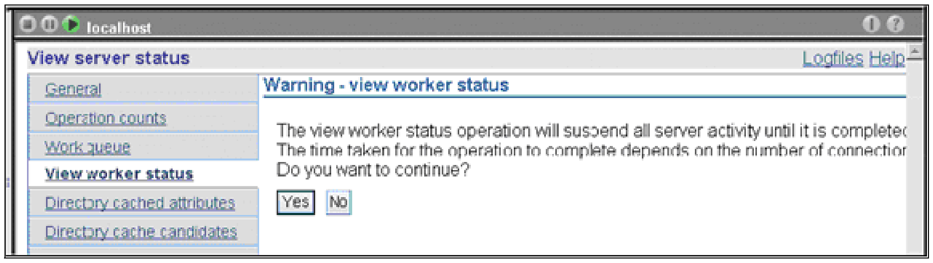
General

Hostname	localhost
Server status	Running
Start time	2004-02-19 03:27:51 GMT
Current time	2004-02-19 07:25:30 GMT
Total threads	1
Total threads blocked on write	0
Total threads blocked on read	0
Number of connections	1

Figure F-1 Viewing the server status

- View the worker thread status

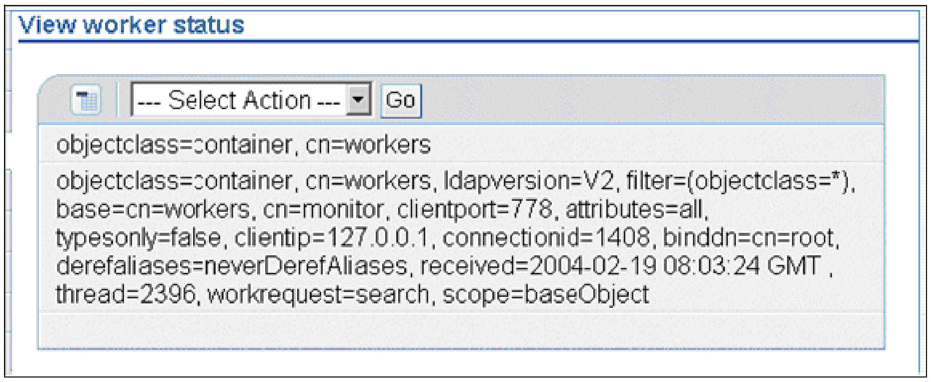
  The state of a worker thread includes many details such as thread number, information about the client it is serving, the type of work request received, and so on. Performing this activity suspends all the server activity until it is completed. see Figure F-2 and Figure F-3.



Warning - view worker status

The view worker status operation will suspend all server activity until it is completec
The time taken for the operation to complete depends on the number of connectior
Do you want to continue?

Figure F-2 Viewing the worker thread status



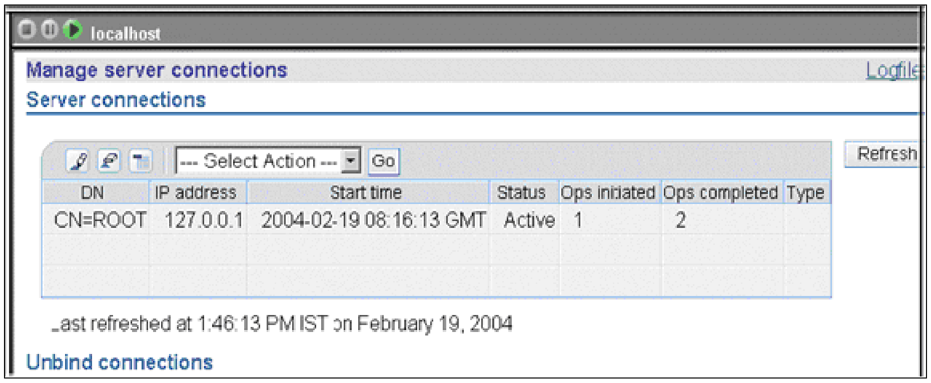
objectclass=container, cn=workers

objectclass=container, cn=workers, ldapversion=V2, filter=(objectclass=*), base=cn=workers, cn=monitor, clientport=778, attributes=all, typesonly=false, clientip=127.0.0.1, connectionid=1408, binddn=cn=root, derefaliases=neverDerefAliases, received=2004-02-19 08:03:24 GMT , thread=2396, workrequest=search, scope=baseObject

Figure F-3 Viewing the worker thread status (continued)

- View connections

  See Figure F-4.



DN	IP address	Start time	Status	Ops initiated	Ops completed	Type
CN=ROOT	127.0.0.1	2004-02-19 08:16:13 GMT	Active	1	2	

Last refreshed at 1:46:13 PM IST on February 19, 2004

Figure F-4 View connections

- View other server status

  a. Connect to the required directory server, using the Web administration tool.
  b. Click **View server status**. This panel has nine tabs.

Command line

You can use the command line to:

- View server status

  The following command returns the state of the server:

  ```
  ibmdirctl-D <adminDN> -w <adminPW> status
  ```

  The above command does not say whether the server is running in safe mode or not. We have to confirm it by running a rootDSE search after this command and check for the attribute ibm-slapdisconfigurationmode, which should be false for a normal mode.

  On UNIX:

  ```
  ldapsearch -D <adminDN> -w <adminPW> -s base objectclass=* | grep config
  ```

- View worker thread status

  View worker thread status using the following command lines.

In order to retrieve all information related to worker threads that are currently active, issue the following command:

```
ldapsearch -D <adminDN> -w <adminPW> -s base -b cn=workers, cn=monitor
objectclass=*
```

You can see an output similar to the one shown in Example F-1.

Example: F-1 Viewing worker thread status

```
cn=workers, cn=monitor
cn=workers
objectclass=container
cn=thread2428, cn=workers, cn=monitor
thread=2428
ldapversion=V2
binddn=cn=root
clientip=127.0.0.1
clientport=2058
connectionid=1412
received=2004-02-19 08:07:41 GMT
workrequest=search
base=cn=workers, cn=monitor
scope=baseObject
derefaliases=neverDerefAliases
typesonly=false
filter=(objectclass=*)
attributes=all
```

- View connections

  We can run a search with the searchbase "cn=connections, cn=monitor" to get information about server connections:

  ```
  ldapsearch -D <adminDN> -w <adminPW> -s base -b cn=connections, cn=monitor
  objectclass=*
  ```

  This command returns information in the following format:

  ```
  cn=connections, cn=monitor
  connection=3 : 127.0.0.1 : 2004-02-22 06:08:10 GMT : 1 : 1 : CN=ROOT : :
  ```

  To end server connections, issue one of the following commands:

  – To disconnect a specific DN:

    ```
    ldapexop -D<adminDN> -w <adminPW> -op unbind -dn cn=john
    ```

  – To disconnect a specific IP address:

    ```
    ldapexop -D <adminDN> -w <adminPW> -op unbind -ip 9.182.173.43
    ```

  – To disconnect a specific DN over a specific IP address:

    ```
    ldapexop -D <adminDN> -w <adminPW> -op unbind -dn cn=john -ip 9.182.173.43
    ```

  – To disconnect all connections:

    ```
    ldapexop -D <adminDN> -w <adminPW> -op unbind -all
    ```

- View other server information

  Using a ldapsearch against base cn=monitor. The command for doing a monitor search is:

  ```
  ldapsearch -D <adminDN> -w <adminPW> -s base -b cn=monitor objectclass=*
  ```

Analyzing log files

The simplest way to get to a problem is to know the time when it has occurred. The log files are time stamped. Therefore, you just compare the different log files simultaneously for the activities at a given instant of time and you are very close to the problem cause. If multiple LDAP servers are involved (for example, debugging a replication issue), keeping them time synchronized is handy (only, of course, if time synchronization is feasible).

Notes:

- Turning on any sort of logging or using a database to log the changes hampers the directory performance. The obvious reason is that such activities make the directory do more things than it liked to. For example, the directory server might have to write to four places rather than one. Hence, it is strictly advised that such options be turned on only in the event that the directory server is doing badly and it needs to be tuned. Disable these options, when you are done with your problem analysis.
- Enabling the change log is seen as a performance bottleneck, because the directory server has to write to the LDAP database as well as log the relevant information in the change log database. Therefore, it is advisable to have the change log enabled only in the event that a problem is being debugged or if another application in your organization (that is, a meta-directory tool) required it to be on.
- Increasing the amount of diagnostic output can result in both performance degradation and insufficient storage conditions in your database instance file system. This procedure should only be used when troubleshooting problems requires the additional diagnostics.

The following sections explains some typical steps for monitoring Tivoli Directory Server status and troubleshooting the problems if any.

Web administration tool

In the Web administration tool, the logfiles field in the task title bar accesses the Web administration console log files.

ibmslapd.log

When a problem occurs that appears to be related to the IBM Directory Server, you should first check the following files for error messages.

The default location is /var/ldap for both Solaris and AIX.

- ibmslapd.log
- db2cli.log

You can change the location of both of these files by modifying the ibm-slapdErrorLog and ibm-slapdCLIErrors parameters in the ibmslapd.conf.

To view the error log, issue the following command (on UNIX):

```
more /var/ldap/ibmslapd.log
```

Where /var/ldap/ibmslapd.log is the default path for the ibmslapd error log.

To view and clear the error log dynamically:

```
ldapexop -D <adminDN> -w <adminPW> -op readlog -log slapd -lines all
ldapexop -D <adminDN> -w <adminPW> -op clearlog -log slapd
```

ibmslapd trace

An ibmslapd trace provides a list of the SQL commands issued to the DB2 database. These commands can help you identify operations that are taking a long time to complete. This information can in turn lead you to missing indexes or unusual directory topology. To turn the ibmslapd trace on, run the following commands:

```
ldtrc on
ibmslapd -h 4096
```

After you have turned the trace on, run the commands that you think might be giving you trouble. Running a trace on several operations can slow the performance, therefore remember to turn the trace off when you finish using it:

```
ldtrc off
```

Change the diagnostic level for error message log files.

DB2 error log

On AIX systems or Solaris operating environments, the db2diag.log file is located, by default, in the /INSTHOME/sqllib/db2dump directory, where INSTHOME is the home directory of the instance owner.

To view the DB2 error log, issue the following command (on UNIX):

```
more /var/ldap/db2cli.log
```

Where var/ldap/db2cli.log is the default path for the DB2 error log.

To view and clear the DB2 error log dynamically:

```
ldapexop -D <adminDN> -w <adminPW> -op readlog -log cli -lines all
ldapexop -D <adminDN> -w <adminPW> -op clearlog -log cli
```

adminAudit.log

/var/ldap/adminAudit.log is the default administration daemon audit log for UNIX systems.

audit.log

/var/ldap/audit.log is the default administration audit log for UNIX systems. The audit.log is disabled by default. You have to enable it either using Web administration tool or command line.

To view the audit log through the command line, issue the following command

(for UNIX):

```
more /var/ldap/audit.log
```

Where /var/ldap/audit.log is the default path for the audit log.

To view and clear the audit log dynamically:

```
ldapexop -D <adminDN> -w <adminPW> -op readlog -log audit -lines all
ldapexop -D <adminDN> -w <adminPW> -op clearlog -log audit
```

Monitoring Tivoli Directory Server performance

Understanding what the directory is doing (and how well it is doing it) is necessary before, during, and after any tuning exercise. Understanding the transaction types, their distribution, and an understanding of the current performance of the directory is necessary to determine what (if any) tuning might be necessary.

The ITDSAUDIT.JAR (covered in 6.1, “ITDSAUDIT.JAR” on page 40) provides a snapshot look at the directory and the operations executed against it by parsing and reporting the transactions as seen by the directory audit log.

Using the ldapsearch utility for monitoring

The `ldapsearch` command can be used to monitor performance, as shown in the following sections.

A number of upgrades to the `cn=monitor` command allows it to pull out more data to better monitor how the LDAP is doing. The monitor search returns some of the following attributes of the server:

cn=monitor

version:	IBM Tivoli Directory, Version 5.2
total connections:	The total number of connections since the server was started
current connections:	The number of active connections
maxconnections:	The maximum number of active connections allowed
writewaiters:	The number of threads sending data back to the client
readwaiters:	The number of threads reading data from the client
opsinitiated:	The number of initiated requests since the server was started
livethreads:	The number of worker threads being used by the server
opscompleted:	The number of completed requests since the server was started
entriessent:	The number of entries sent by the server since the server was started
searchesrequested:	The number of initiated searches since the server was started
searchescompleted:	The number of completed searches since the server was started
filter_cache_size:	The maximum number of filters allowed in the cache
filter_cache_current:	The number of filters currently in the cache
filter_cache_click:	The number of filters retrieved from the cache rather than being resolved in DB2
filter_cache_miss:	The number of filters that were not found in the cache that then needed to be resolved by DB2
filter_cache_bypass_limit:	Search filters that return more entries than this limit are not cached
entry_cache_size:	The maximum number of entries allowed in the cache
entry_cache_current:	The number of entries currently in the cache
entry_cache_click:	The number of entries that were retrieved from the cache
entry_cache_miss:	The number of entries that were not found in the cache that then needed to be retrieved from DB2
acl_cache:	A Boolean value indicating that the ACL cache is active (TRUE) or inactive (FALSE)

acl_cache_size:	The maximum number of entries in the ACL cache
currenttime:	The current time on the server. The current time is in the format: year month day hour:minutes:seconds GMT.
starttime:	The time the server was started. The start time is in the format: year month day hour:minutes:seconds GMT.
en_currentregs:	The current number of client registrations for event notification
en_notificationssent:	The total number of event notifications sent to clients since the server was started

The following attributes are used for operation counts:

bindsrequested:	The number of bind operations requested since the server was started
bindscompleted:	The number of bind operations completed since the server was started
unbindsrequested:	The number of unbind operations requested since the server was started
unbindscompleted:	The number of unbind operations completed since the server was started.
addsrequested:	The number of add operations requested since the server was started
addscompleted:	The number of add operations completed since the server was started
deletesrequested:	The number of delete operations requested since the server was started
deletescompleted:	The number of delete operations completed since the server was started
modrdnsrequested:	The number of modify RDN operations requested since the server was started
modrdnscompleted:	The number of modify RDN operations completed since the server was started
modifiesrequested:	The number of modify operations requested since the server was started
modifiescompleted:	The number of modify operations completed since the server was started
comparesrequested:	The number of compare operations requested since the server was started
comparescompleted:	The number of compare operations completed since the server was started
abandonsrequested:	The number of abandon operations requested since the server was started
abandonscompleted:	The number of abandon operations completed since the server was started
extopsrequested:	The number of extended operations requested since the server was started
extopscompleted:	The number of extended operations completed since the server was started

unknownopsrequested:	The number of unknown operations requested since the server was started
unknownopscompleted:	The number of unknown operations completed since the server was started

The following attributes are used for server logging counts:

slapderrorlog_messages:	The number of server error messages recorded since the server was started or since a reset was performed
slapdclierrors_messages:	The number of DB2 error messages recorded since the server was started or since a reset was performed
auditlog_messages:	The number of audit messages recorded since the server was started or since a reset was performed
auditlog_failedop_messages:	The number of failed operation messages recorded since the server was started or since a reset was performed

The following attributes are used for connection type counts:

total_ssl_connections:	The total number of Secure Sockets Layer (SSL) connections since the server was started
total_tls_connections:	The total number of Transport Layer Security (TLS) connections since the server was started

The following attributes are used for tracing:

trace_enabled:	The current trace value for the server. TRUE, if collecting trace data; FALSE, if not collecting trace data
trace_message_level:	The current ldap_debug value for the server. The value is in hexadecimal form, for example: - 0x0=0 - 0xffff=65535
trace_message_log:	The current LDAP_DEBUG_FILE environment variable setting for the server

The following attributes are used for denial of service prevention:

available_workers:	The number of worker threads available for work
current_workqueue_size:	The current depth of the work queue. largest_workqueue_size: The largest size that the work queue has ever reached.
idle_connections_closed:	The number of idle connections closed by the Automatic Connection Cleaner
auto_connection_cleaner_run:	The number of times that the Automatic Connection Cleaner has run
emergency_thread_running:	The indicator of whether the emergency thread is running
totaltimes_emergency_thread_run:	The number of times the emergency thread has been activated
lasttime_emergency_thread_run:	The last time the emergency thread was activated

The following attribute has been added for alias dereference processing:

bypass_deref_aliases:	The server runtime value that indicates if alias processing can be bypassed. It displays TRUE if no alias object exists in the directory, and FALSE if at least one alias object exists in the directory.

The following attributes are used for the attribute cache:

cached_attribute_total_size:	The amount of memory used by the directory attribute cache, in KB. This number includes additional memory used to manage the cache that is not charged to the individual attribute caches. Consequently, this total is larger than the sum of the memory used by all the individual attribute caches.
cached_attribute_configured_size:	The maximum amount of memory, in KB, assigned to the directory attribute cache
cached_attribute_click:	The number of times the attribute has been used in a filter that could be processed by the attribute cache. The value is reported as follows: cached_attribute_click=attrname:#####
cached_attribute_size:	The amount of memory used for this attribute in the attribute cache. This value is reported in KB as follows: cached_attribute_size=attrname:######
cached_attribute_candidate_click:	A list of up to ten most frequently used noncached attributes that have been used in a filter that could have been processed by the directory attribute cache if all of the attributes used in the filter had been cached. The value is reported as follows: cached_attribute_candidate_click=attrname:#####

G

Additional material

This IBM Redpaper refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this IBM Redpaper is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/REDP4258

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select the **Additional materials** and open the directory that corresponds with the IBM Redpaper form number, REDP4258.

Using the Web material

The additional Web material that accompanies this IBM Redpaper includes the following files:

Tools_Sample_Data_and_Outputs.tar

This tar file contains a lot of scripts, LDAP Data Interchange Format (LDIF) files, spreadsheets, and other documents that are used and referenced throughout the IBM Redpaper.

How to use the Web material

Create a subdirectory (folder) on your workstation, and untar the contents of the Web material zip file into this folder.

Related publications

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

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 196. Note that some of the documents referenced here may be available in softcopy only.

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

Other publications

These publications are also relevant as further information sources:

- *IBM Tivoli Directory Server Administration Guide Version 6.0*, SC32-1674
- *IBM Tivoli Directory Server Problem Determination Guide Version 6.0*, SC32-1679
- *IBM DB2 Universal Database Administration Guide: Implementation Version 8*, SC09-4820
- *IBM DB2 Universal Database Administration Guide: Performance Version 8*, SC09-4821
- *IBM DB2 Universal Database Administration Guide: Planning Version 8*, SC09-4822
- *IBM DB2 Universal Database Command Reference Version 8*, SC09-4828
- *IBM Advanced DBA Certification Guide and Reference for DB2 Universal Database v8 for Linux, UNIX, and Windows* by Dwaine R. Snow and Thomas X. Phan

Online resources

These Web sites are also relevant as further information sources:

- IBM Tivoli Directory Server online resource for product manuals

  http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?toc=/com.ibm.IBMDS.doc/toc.xml
- DB2 9 for Linux UNIX and Windows

  http://www.ibm.com/software/data/db2/udb/support/
- IBM DB2/UDB v8 online resource for product manuals

  http://www.ibm.com/software/data/db2/udb/support/manualsv8.html
- DB2 Information Center site:

  http://publib.boulder.ibm.com/infocenter/db2help/index.jsp
- DB2 DeveloperWorks site:

  http://www.ibm.com/developerworks/db2/

- DB2 library

  http://www.ibm.com/software/data/db2/library
- IBM Problem Support 1-800-IBM-SERV

  http://www.ibm.com/software/support/probsub.html
- Fix pack download site:

  http://www.ibm.com/software/data/db2/udb/support/downloadv8.html
- AIX Toolbox for Linux Applications

  http://www.ibm.com/servers/aix/products/aixos/linux/rpmgroups.html
- *Disabling replication conflict resolution*

  http://www.ibm.com/support/docview.wss?rs=767&context=SSVJJU&q1=conflict+resolution&uid=swg21236775&loc=en_US&cs=utf-8&lang=en

How to get IBM Redbooks

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

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

Performance Tuning for IBM Tivoli Directory Server

Redpaper

Performance tuning for Tivoli Directory Server for very large user environments

Complete coverage from operating system to database tuning

Extensive scripts and checklists

In today's highly connected world, directory servers are the IT cornerstone of many businesses. These components of the corporate infrastructure are the foundation of authentication systems for internal, and more commonly, external user populations. Managing a directory server with several hundred internal users is not all that difficult. However, when managing a directory server with several million external users in all 24 time zones throughout the world is a much more daunting task.

IBM Tivoli Directory Server is capable of handling millions of entries given the right architecture, configuration, and performance tuning—tunings that can differ greatly from that of a smaller server with only a few hundred thousand entries. Managing and tuning a directory server of this size requires a change in mindset: No longer can tuning be done after the fact. Tuning and performance must be a focus before the hardware is even ordered. A proactive role must be taken after installation as well, including pre-tuning steps to better interface with other products to make installations and migrations more successful, and regular maintenance to keep the directory well tuned and running smoothly.

This IBM Redpaper is the cumulation of lessons learned in many different real-world environments, including a 24-server fault tolerant configuration with over 300 million entries. The authors have pooled their collective knowledge and resources to provide the most comprehensive performance view possible, from hardware to software, sort heaps to buffer pools, and table cardinalities to explain plans. In large directory server deployments, use this document as an outline on how to get the right fit for your environment.

INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

For more information:
ibm.com/redbooks