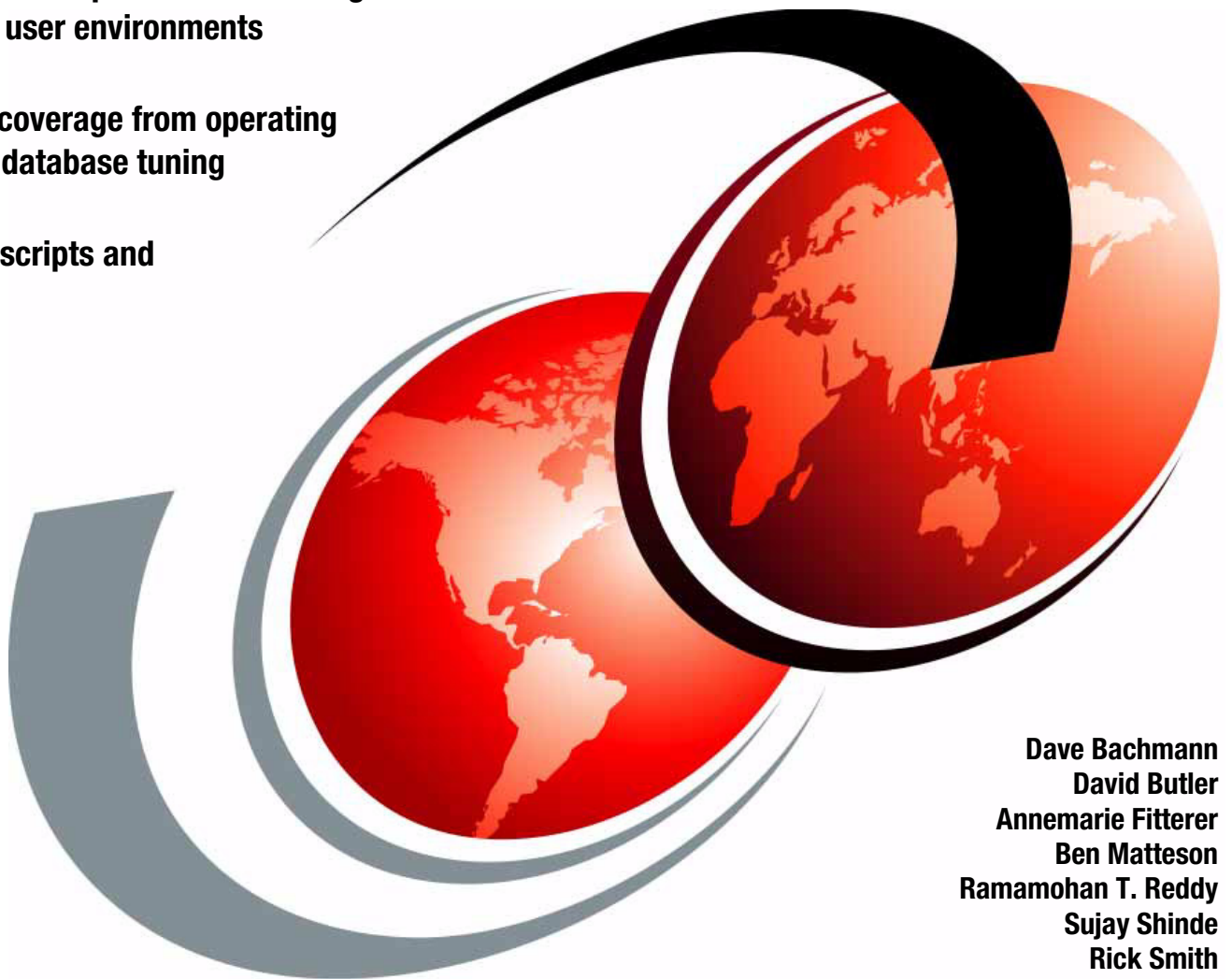


Performance Tuning for IBM Security Directory Server

Directory Server performance tuning for
very large user environments

Complete coverage from operating
system to database tuning

Extensive scripts and
checklists



Dave Bachmann
David Butler
Annemarie Fitterer
Ben Matteson
Ramamohan T. Reddy
Sujay Shinde
Rick Smith



International Technical Support Organization

Performance Tuning for IBM Security Directory Server

April 2014

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

Second Edition (April 2014)

This edition applies to different versions of IBM Security Directory Server and its underlying support software. Be aware that with v6.3.1 the name of the product changed from IBM Tivoli Directory Server to IBM Security Directory Server. In this book we mostly refer to *Directory Server* when the version is of no importance.

© Copyright International Business Machines Corporation 2014. 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

Notices	ix
Trademarks	x
Preface	xi
Authors	xi
Become a published author	xiii
Comments welcome	xiii
Chapter 1. Service level objectives and agreements	1
1.1 Common service level agreements and objectives	2
1.1.1 SLA and SLO guiding principles	2
1.1.2 Lightweight Directory Access Protocol specific SLAs	2
1.2 When to contact IBM support services	3
Chapter 2. Designing your directory for optimal performance	5
2.1 Strategies for defining a new directory	6
2.1.1 Performance considerations	6
2.1.2 SMS versus DMS table space considerations	7
2.1.3 Static versus dynamic groups	7
2.2 Access control lists and their impact	9
2.3 Anticipating growth	10
2.3.1 Data considerations	10
2.3.2 Workload considerations	10
Chapter 3. Time to do a health check	11
3.1 Questions to ask	12
3.2 Diagrams and layouts of the system	12
3.3 Configurations, logs, and outputs	12
3.4 Scripts to help gather information	13
3.4.1 perfcheck_database.sh	14
3.4.2 perfcheck_runstats.sh	14
3.4.3 perfcheck_system.sh	15
3.4.4 perfcheck_ldap.sh	15
3.5 IBM DB2 monitors	16
3.6 Analyzing the gathered information	17
Chapter 4. Tips for tuning DB2 performance	19
4.1 Tuning DB2 and Directory Server caches and parameters	20
4.1.1 Determine the number of entries and average entry size with idsperftune . . .	20
4.1.2 Basic tuning	21
4.1.3 Advanced tuning	23
4.2 Database maintenance utility: idsdbmaint	25
4.2.1 Index reorganization	26
4.2.2 Row compression	27
4.2.3 Table space conversion	28
4.3 DB2 self-tuning memory	30
4.4 More tuning resources	31
Chapter 5. DB2 settings related to LDAP	33

5.1 IBM DB2 self-tuning memory overview	34
5.2 Adjusting the buffer pool and sort heap threshold settings	34
5.3 Self-tuning memory and the sortheap parameter	34
5.4 Self-tuning memory and the pckcachesz parameter	35
5.5 Self-tuning memory and the locklist parameter	35
5.6 Self-tuning memory and health checks	35
5.7 DB2SET commands	36
5.7.1 DB2_PARALLEL_IO	36
5.7.2 DB2_HASH_JOIN	36
5.7.3 DB2_ANTIJOIN	36
Chapter 6. Using LDAP_MAXCARD and IBMSLDAP_USE_SELECTIVITY	37
6.1 Table cardinality and performance	38
6.1.1 Setting DB2 table statistics	40
6.2 LDAP_MAXCARD setting	40
6.3 LDAP and DB2 Selectivity	41
6.4 Additional indexes	42
Chapter 7. Tools and scripts	43
7.1 ITDSAUDIT.JAR	44
7.1.1 Theory of operation	44
7.1.2 Prerequisites	44
7.1.3 Starting itdsaudit.jar application	45
7.1.4 itdsaudit.jar error messages	45
7.1.5 itdsaudit.jar stdout output	45
7.1.6 itdsaudit.jar PDF output	46
7.2 tune_enablemonitor.sh	53
7.3 perftune_enablemonitor_all.sh	53
7.4 tune_disablemonitor.sh	54
7.5 perfanalyze_indexes.pl	54
7.5.1 Use	54
7.5.2 Examples	55
7.6 perfanalyze_audit.pl	55
7.6.1 Use	55
7.6.2 Examples	56
7.7 perfanalyze_dynamicsql.pl	56
7.7.1 Use	57
7.7.2 Examples	57
7.8 perfanalyze_database.pl	57
7.8.1 Use	58
7.8.2 Examples	58
7.9 perfanalyze_tables.pl	58
7.9.1 Use	58
7.9.2 Examples	59
Chapter 8. Why you must use runstats	61
8.1 Optimization	62
8.2 Which runstats to use	62
8.3 DB2 automatic statistics collection	63
Chapter 9. When and how to run reorg	67
9.1 Performing a reorg when necessary	68
9.1.1 Performing a reorgchk	68
9.1.2 Reorganize a table	70

9.1.3 Reorganize an index	71
9.1.4 Guidelines for performing a reorganization	72
9.2 DB2 automatic table and index reorganization	73
9.2.1 Parameter description	73
Chapter 10. LDAP searches and slow operations	75
10.1 Improving LDAP searches	76
10.2 Identifying slow operations	77
10.2.1 Auditing for performance	77
10.2.2 Tracing performance	79
Chapter 11. Indexes and direct I/O operations	81
11.1 Indexes explained	82
11.1.1 Optimizing indexes by using DB2 commands	82
11.1.2 Optimizing searches by using DB2 explain	83
11.2 Direct I/O	86
Chapter 12. Disk striping and RAID	89
12.1 Considerations for RAID arrays	90
Chapter 13. Distributed directory	91
13.1 Distributed directory environment	92
13.1.1 The need for distributed directories	92
13.1.2 Distributed directory and proxy server	92
13.1.3 Terminology related to a distributed directory and proxy server	92
13.1.4 Configuring a distributed directory environment	94
13.1.5 A closer look at proxy server configuration	95
13.2 Strategies for partitioning data	98
13.2.1 RDN hash-based splitting	98
13.2.2 Subtree-based splitting	102
13.2.3 Hybrid splits	107
13.2.4 Nested splits	112
13.3 Fail over and load balancing	118
13.3.1 Replicated back-ends	121
13.4 Tuning the proxy server	125
13.4.1 Front-end environment variables	125
13.4.2 Tuning attributes in Proxy Backend database entry	128
13.4.3 Tuning attributes in Backend server entry	130
13.4.4 Tuning attributes in partition base entry	131
13.4.5 Tuning attributes in partition entry	132
Chapter 14. LDAP replication information	135
14.1 Replication performance considerations	136
14.1.1 Cryptographic synchronization	136
14.1.2 Replication by using SSL or not	136
14.1.3 Conflict resolution	137
14.1.4 Multithreaded (asynchronous) or single-threaded replication	137
14.2 The importance of cn=ibmpolicies replication	138
14.3 Distinguishing between LDAP reads and writes	139
14.4 Conflict resolution	140
14.5 Monitoring and managing replication	141
14.5.1 Operational attributes	142
14.5.2 Environment variables for use with replication	143
14.5.3 Troubleshooting replication problems	144

14.6 Synchronizing two-way cryptography for server instances	144
Chapter 15. Adding a new LDAP server to an existing enclave	147
15.1 Installing a new Directory Server	148
15.1.1 Update the server ID in the LDAP server configuration	148
15.2 Adding a new LDAP server to replication topology	149
15.2.1 Define the role of the new LDAP server	149
15.2.2 Define the default credential object and referral that is based on the role	149
15.2.3 Create the replication settings to add the new server.	150
15.2.4 Load the new agreement	151
15.2.5 Back up data from a Directory Server v6 peer master server.	152
15.2.6 Restore data to the new LDAP server.	152
15.2.7 Start all new LDAP servers and verifying replication queues	153
15.3 Using the idsideploy method to add a new LDAP server	154
15.3.1 Prepare an existing peer (source) server	154
15.3.2 Use idsideploy on the second (target) server	154
15.4 Testing replication	158
Appendix A. Special operating system tuning for IBM Directory Server.	161
Oracle Solaris and HP-UX operating system tuning	162
Determining which system settings are required for DB2 and LDAP	162
Linux operating system tuning	163
To update kernel parameters on Red Hat and SUSE Linux	163
IBM AIX operating system tuning.	165
Enabling large files	165
Setting MALLOCOPTIONS.	165
Setting other environment variables	166
Viewing ibmslapd environment variables on AIX	171
Appendix B. How to apply fix packs to an LDAP server	173
Appendix C. IBM DB2 UDB concepts and definitions	175
Appendix D. DB2 UDB quick reference guide	179
DB2 command-line processor	180
Instance configuration	180
Instance configuration keywords	180
DB2 registry configuration	181
Catalog remote database.	181
DB2 instance start and stop.	181
Database commands	181
Database connection	182
Database creation	182
Database object display.	183
Database configuration	183
Database privileges	183
Database statistics updates	184
DB2 database monitoring.	184
Database recovery	184
Troubleshooting	185
Appendix E. Directory Server backup and restore methods	187
DB2 information	188
Directory schema and database definitions	188

Directory Server V6.x directory schema	188
Directory Server V6.x directory database definitions	189
Directory Server directory database and table spaces	189
Directory Server change log database and table spaces	191
Distributing databases across multiple physical disks	191
Creating file systems and directories on the target disks	191
Backing up the existing database	192
Performing a redirected restore of the database	193
Overview of offline backup and restore procedures for LDAP	195
Replication considerations	196
Overview of online backup and restore procedures for LDAP	196
Offline and online backup and restore examples	198
Example DB2 list history information	198
Example of an offline backup and restore procedures	199
Example of an online backup for the directory database	200
Incremental directory and change log database online backup	201
Creating full offline backups for directory and change log databases	202
Creating incremental online backups for directory and change log databases	202
Restoring the directory database	203
Restoring both directory and change log databases	204
Using incremental delta backups	205
Restoring from incremental delta backups	205
Pros and cons of different recovery strategies	206
Other backup, restore, and rollforward command options	206
Common problems for backup, restore, rollforward commands	207
Optional migration from V5.2 to V6.0 to enable online backup	208
Evaluating BLOB columns on Directory Server 5.2 and 6.0	209
blobmigrate1G script	214
Appendix F. Checklists	225
Maintenance checklist	226
IBM Security Directory Server support tool	226
Monitoring checklist	226
Key reasons for monitoring	227
Monitoring for outages and performance degradations	228
Monitoring performance and service level agreement conformance	229
Monitoring Directory Server status	229
Using the ldapsearch utility for monitoring	232
Analyzing log files	236
Monitoring Directory Server performance	237
.	238
Appendix G. Additional material	239
Where to find the web material	240
How to use the web material	240
Related publications	241
IBM Redbooks	241
Other publications	241
Online resources	241
How to get IBM Redbooks publications	242
Help from IBM	242

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

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. These and other IBM trademarked terms are marked on their first occurrence in this information with the appropriate symbol (® or ™), indicating US registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at <http://www.ibm.com/legal/copytrade.shtml>

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

AIX®
DB2®
DB2 Universal Database™
developerWorks®
Guardium®

IBM®
InfoSphere®
RDN®
Redbooks®
Redpaper™

Redbooks (logo) ®
Tivoli®
WebSphere®
z/OS®

The following terms are trademarks of other companies:

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

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

Java, and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

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, managing a directory server with several million external users in all 24 time zones throughout the world is a much more daunting task.

IBM® Security Directory Server software can handle millions of entries, given the right architecture, configuration, and performance tuning. However, that tuning can differ greatly from tuning for a smaller server with only a few hundred thousand entries. Managing and tuning a directory server of this size requires a change in mindset. Tuning and performance must be a focus even before the hardware is ordered. A proactive approach must be taken after installation also, including the pretuning steps to better interface with other products to make installations and migrations successful, and then regular maintenance to keep the directory running smoothly.

This IBM Redbooks® publication is the accumulation of lessons learned in many different real-world environments, including a 24-server fault tolerant configuration with more than 300 million entries. The authors pooled their knowledge and resources to provide the most comprehensive performance view possible, from hardware to software, sort heaps to buffer pools, and table cardinalities.

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

Name change: This edition applies to different versions of IBM Security Directory Server and its underlying support software. With Version 6.3.1, the name of the product changed from IBM Tivoli® Directory Server to IBM Security Directory Server. In this book, we call it just “Directory Server” when the version not relevant.

Some of the screen captures still show IBM Tivoli Directory Server in the title, but the same functions are accessible in IBM Security Directory Server. That is also true for some of the referenced external documentation.

Authors

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

Dave Bachmann is a Senior Technical Staff Member who leads the performance work for IBM Security Systems Identity and Access products. He is also the Directory Chief Architect. Dave has worked on the performance of distributed systems for the last 26 years. He has been involved with the performance of IBM Security Directory Server throughout the product's existence.

David Butler is a Senior Program Manager and Team Lead in the IBM Security Services Division. He has more than 35 years of experience in the information technology field, with the last 15 years spent working on the Tivoli suite of products, specifically security. Recently, he was the Security Engineer and Lead for the internal IBM CIO InfoSphere® Guardium®

rollout. Dave holds a software patent (IBM) and has collaborated on and produced various technical papers relevant to systems and information management.

Annemarie Fitterer is a manager with IBM WebSphere® Application Server Distributed L2 support. She has over 10 years experience with the Directory Server, including roles in testing, development, and support. From mid-2006 to mid-2013, she acted as the Directory Server L2 Support Team Lead. She has worked directly with several large client deployments and partnered with Sales and Service to ensure the success of IBM clients and Business Partners. Annemarie has multiple patents and has written, contributed to, and presented several highly valued Directory Server Support Technical Exchanges through the years.

Ben Matteson is an Advisory Software Support Engineer who has been working for IBM Tivoli Support in Austin, Texas, since 1997. He has worked with the Tivoli Framework, Tivoli Distributed Monitoring, Tivoli Monitoring 5.x (including the original Tivoli Data Warehouse 1.x product) and 6.x products, and has been on the Support team for IBM Security Directory Server since 2005, supporting its many enterprise users. Ben has expertise in Directory Server, Linux, IBM DB2®, Perl, SQL, and TCP/IP networking.

Ramamohan T. Reddy is a Senior Software Support Engineer who works for IBM Security Support. He has 20 years of experience with system software design, development, testing, and support. He has worked with IBM Security Directory Server since 2001, with the last eight years in support. Ram has working knowledge of several security products, including LDAP servers from different vendors and is an IBM Certified Deployment Professional for Tivoli Directory Server V6.3. He holds a Master's degree in Electronics and Communication Engineering and an IBM certification for software development.

Sujay Shinde works with the IBM Security Privileged Identity Manager virtual appliance team. He has been involved with Directory Server development for more than seven years to deliver features in the performance and scalability areas. He worked to certify IBM Security Directory Server for Common Criteria at EAL4 level and to comply with NIST SP800-131A and FIPS 140-2 security standards.

Rick Smith is a developer who works in the L3 support group for the IBM directory product since the turn of the millennium. Rick has several patents in the IBM directory and other areas, some with worldwide grants. He has also spent time in software testing, including making presentations at national conferences.

We thank the following people who contributed to this paper:

Brook Heimbaugh, Directory Server L3 Team Lead, IBM Austin
Robert D. Hodges, Systems Management Consultant, IBM Raleigh

Thanks to the authors of the first edition of this IBM Redbooks publication, *Performance Tuning for IBM Tivoli Directory Server*, REDP-4258, published in March 2007:

Robert Hodges
Richard Macbeth
John McGarvey
Casey Peel
Jukka Rantanen

We also thank the following people for their involvement with the first edition of this paper, all of whom work for IBM USA unless otherwise indicated:

Shirley Chrisco, Software Engineer, Information Development, Software Group, Tivoli
Cindy Corn, Software Engineer, LDAP Development, Software Group, Tivoli
Jeff DeMent, Tivoli Directory, Security Architect, Software Group, Americas Security Practice
Karen Duxbury, IBM Canada, IBM Toronto Lab
Shevaun M. Fontenot, Software Engineer, LDAP Development, Software Group, Tivoli
Anne Lesell, IBM Finland, IT Specialist, DB2 Information Technology Services
Mark McConaughy, Software Engineer, LDAP Development, Software Group, Tivoli
Michael Seedorff, IT Specialist, IBM Software Group, Americas Security Practice
Ram Sreerangam, Security Consultant, Software Group, Americas Security Practice
John D. Sullivan, Software Engineer, Security Architect, Software Group, Tivoli
Chunhui Yang, Certified Sir ITS, Software Group, 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 Redbooks publication that deals 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:

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 publication in one of the following ways:

- Use the online **Contact us** review form:

ibm.com/redbooks

- Send your comments by email:

redbooks@us.ibm.com

- Mail your comments:

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



Service level objectives and agreements

Every component within an IT environment needs 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 can 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 whether it is still meeting the objectives and agreements.

With IBM Security Directory Server, like many applications that are transaction based, it can be difficult to predict the operation and performance on paper. Until it is configured, the data is loaded, and the actual client transactions are applied, only general guidelines 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 the time span of using the Directory Server.

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

For these reasons, we recommend that all instances of IBM Security Directory Server be benchmarked. Then, after you define SLAs or SLOs, we advise that you take operational measurements regularly to ensure that it is meeting the performance objectives that you set.

1.1 Common service level agreements and objectives

Consider the following general guidelines and Lightweight Directory Access Protocol (LDAP) specific details.

1.1.1 SLA and SLO guiding principles

To be meaningful, service level agreements or objectives (SLA or 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, objective values.
- ▶ Items that are not under the direct control of the application or entity (for example, network transition time) fall under the heading of an end-to-end SLA or SLO, for which there should be an individual component part SLAs or SLOs.
- ▶ The SLA or SLO must be reasonable and attainable with the chosen design and topology.

1.1.2 Lightweight Directory Access Protocol specific SLAs

For the IBM Security Directory Server, all SLA/SLO measurements are defined as the average time taken for internal processing and the start of data delivery. By measuring only the time to process the operation (bind, add, modify, search, and so on), you 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 use.

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 execution times

The average times for an SLA are measured in milliseconds from the receipt of a correctly formatted and valid operation to the initial return of a successful reply to the client application. That time includes these operations:


- ▶ Average bind time
- ▶ Average unbind time
- ▶ Average add time
- ▶ Average modify time
- ▶ Average delete time
- ▶ Average search time

1.2 When to contact IBM support services

IBM product support teams are not structured to assist with all issues. Their primary focus is to provide support if there is a defect. If you are experiencing a functional problem with the product, engaging the product support team is the best method to pursue resolution.

If you need assistance with the following types of activities, please contact your IBM representative to discuss which advanced support or services can best meet your needs:

- ▶ Performance testing, tuning, and analysis
- ▶ Writing, reviewing, troubleshooting custom code
- ▶ Extensive configuration questions
- ▶ Consulting
- ▶ Migration planning
- ▶ Architectural review
- ▶ Solution design
- ▶ Meeting your service level agreements



Designing your directory for optimal performance

When you are designing your Directory Server environment with performance in mind, there are several things to consider.

2.1 Strategies for defining a new directory

Typically, users interact with the directory only through middleware applications and do not have a choice in how the directory is structured. However, if you are designing a new application and need to decide the best way to structure your directory data, here are some things to consider.

2.1.1 Performance considerations

Groups are a common cause of performance problems because they can be both large and volatile yet also heavily relied upon for searches and access control list (ACL) evaluation. Special strategies for groups are described in 2.3, “Anticipating growth” on page 10.

Workload considerations: read versus write operations

Directories are designed and optimized to allow many users to search and read data simultaneously. But whenever an update is performed, a user must get exclusive access to the data before the update can proceed. The update must wait for all current read operations to finish before it can get access, and then block all pending read operations while it performs the update. If you have data that is updated frequently, there are ways to minimize the impact of those updates on other read traffic:

- Do not index or search volatile attributes.

Attributes that change rarely make better search filters than volatile attributes that change frequently. Updating an attribute prevents other users from searching it, and maintenance tasks for the indexes increase the time that is needed to update the attribute.

- Do not index or search volatile entries.

Moving the volatile attributes into a separate entry allows it to be updated without any impact to reads or searches of the static attributes. For example: An entry that represents a printer contains static values, such as name, model, and location. Those change rarely, if at all, and are ideal targets for indexing and searching. But it is better to store attributes that change frequently, such as the job queue, in a separate sub-entry that can be locked and updated without affecting concurrent searches.

Data considerations

The size and structure of your directory can affect the performance of your environment. When you are defining a directory, it is important to consider the relationship of the data, the layout of your entries, how data will be accessed, and the content of the entries. Data redundancy and failover must also be considered to ensure high availability and opportunities for maintenance.

Directory information tree structure

The first step in defining your directory information tree (DIT) is to consider your intended data. What common elements exist within the data and what are the logical division points?

A suffix or naming context is a distinguished name (DN) that identifies the top or root entry of the directory hierarchy. Name the suffix in a meaningful way to represent the data that it contains. If more logical division points exist, data can be further partitioned into subtrees within the same suffix. If there are no commonalities among the data, you can define more suffixes.

When partitioning data, it is important to group similar data within the same subtree. Organizing your DIT by grouping similar data helps simplify the implementation of access requirements and replication configuration.

Entry size

The Directory Server must allocate memory to store the entries in use. If an entry is less than ~ 20 KB, it fits in the default buffer size. But larger entries require special handling. As with volatile data, moving large attributes into separate subentries that are not involved in routine searches can help.

Replication

When you are designing a DIT, it is important to plan for redundancy and high availability. You can configure replication to meet these requirements. When you are designing your DIT structure and considering what type of replication topology to implement, it is important to partition the data that you plan to replicate within the same subtree.

Access considerations: access control lists

ACL processing can account for a large portion of the work that is involved in an operation. Minimizing and simplifying ACLs is an important performance consideration. It is especially important to avoid ACLs, such as these, that rely on large or volatile groups:

- ▶ Administrator access
Operations that are performed as (or by) the directory administrator bypass most ACL processing.
- ▶ Nonfiltered versus filtered ACLs
Nonfiltered ACLs are usually less expensive to evaluate than filtered ACLs. Partitioning entries into separate subtrees that all share a common ACL definition at the root is a useful strategy.

2.1.2 SMS versus DMS table space considerations

In general, the default configuration for using database-managed space (DMS) table spaces is the best one to use for a Lightweight Directory Access Protocol (LDAP) directory, because of the requirements of large scale and fast performance in searching for a small set of matching entries in a large subtree. You can find an explanation of the performance characteristics of the different types of table spaces in the “SMS and DMS workload considerations” section of the IBM DB2 Information Center:

<http://bit.ly/1oHtrCf>

DMS table spaces with no file system caching are the best choice for high performance and large scale, so that is what is created by default when you create a directory. The limit on the size of a DMS table space is 64 terabytes.

2.1.3 Static versus dynamic groups

When you are determining what kind of group to use, there are several considerations:

- ▶ How large do you expect the group to be?
- ▶ Will the membership of the group change frequently or remain relatively static?
- ▶ How do applications retrieve and use the group information?
- ▶ How do you expect the group to change over time?

Static and dynamic groups are optimized for different requirements:

- ▶ Static LDAP groups are designed for high performance and to scale to millions of members. Checking or modifying a user's group membership involves just a quick lookup or update to the member table. Users are added to or removed from a static group by updating that group.

- Dynamic groups are designed for runtime resolution by evaluation of an LDAP search filter that is associated with that group. Checking a user's group membership means evaluating the search filter that is associated with every known dynamic group. But a user's group membership can be modified only by changing the user's attributes that are referenced by the search filter associated with that group.

Because of this, static groups perform much better than dynamic groups for most operations. Dynamic groups are faster at changing the group membership of a large number of users if that is done by modifying the search filter that is associated with the group.

Preferred practices

Follow these practices for efficiency:

- Avoid group-based password policies. If you have defined group password policies, the performance of bind operations might be degraded. This is because a user's effective password policy must be determined during authentication, and group membership must be resolved to properly apply group password policies.
- Filtered ACLs are more expensive in terms of evaluation. It is more efficient to use nonfiltered ACLs.
- Disable Group Conflict Resolution if the environment is configured for replication.
- Retrieve only the attributes that you need. Do not use ALL by default. For example, when you search for the groups that a user belongs to, ask for only the Distinguished Names (DNs), not the entire group. If possible, do not request the member or uniquemember attributes.

Static groups

Read/write performance for this type of group can be affected by the size of group membership, because each attribute value in the entry is stored as a database (DB) table entry. Also, a check to eliminate duplicate attribute values is performed when you add a member to the group.

Static groups are best for these situations:

- Relatively static membership (adding or removing members modifies the group directly, which can be difficult if it is large and heavily used)
- Large number of relatively small groups (`ibm-allGroups` is very efficient)

Dynamic groups

Read performance for this type of group can be affected by the size of the directory, because group membership is determined by a directory search from a specified base location. Because directory entries are unique, no duplicate check needs to be done.

Dynamic groups are best suited for these situations:

- Highly dynamic membership (adding or removing members modifies only the member attributes, not the group)
- Small number of very large groups (`ibm-allGroups` can be expensive with many dynamic groups)

Nested groups

Write performance for this type of group can be affected by the depth of the nested group hierarchy, because each of the ancestor or descendant relationships in the hierarchy is stored as a DB table entry. Also, checking to eliminate duplicate relationships is performed when child group members are added to the parent group.

When you use nested groups, limit the depth of nesting to 50 groups or fewer. Greater nesting depths can result in greater processing times during add or delete operations that involve updates to the nested group hierarchy.

2.2 Access control lists and their impact

Access control lists (ACLs) provide a way to protect information that is stored in an LDAP directory. Administrators use ACLs to restrict access to different portions of the directory or to specific directory entries. LDAP directory entries are related to each other by a hierarchical tree structure. As mentioned, ACL processing can significantly affect the performance of the Directory Server.

There are two types ACLs:

- ▶ Nonfiltered ACLs

This type of ACL applies specifically to the directory entry where it is stored, but it can be propagated to all or none of its descendant entries.

- ▶ Filtered ACLs

Filter-based ACLs employ a filter-based comparison, by use a specified object filter, to match target objects with the effective access that applies to them.

Although they perform the same function, the behavior of the two types of ACLs is significantly different. Filter-based ACLs do not propagate in the same way that nonfilter-based ACLs currently do. The default behavior of filter-based ACLs is to accumulate upward along the ancestor chain, from the lowest to highest containing entry in the DIT. The effective access is determined by joining the access rights that are granted or denied by the constituent ancestor entries. The only exception to this behavior is if replication is enabled. In that case, a ceiling attribute is used to stop accumulation at the entry in which it is contained.

As a result of the additional processing that is required for filter-based ACLs, it is best to partition similar entries in separate subtrees that share a common ACL definition by using nonfilter based ACLs set at the root of the subtree.

Another strategy to eliminate ACL processing for a specific application is to add the application bind DN as a member of the Global Administrative Group. The global administrative group is a way for the directory administrator to delegate administrative rights in a distributed environment to the database back end. Global Administrative Group members are users who are assigned the same set of privileges as the Administrative Group for accessing entries in the database back end. Members of the Global Administrative Group do not have access to the audit log, and they have no privileges or access rights to any data or operations that are related to the configuration settings (configuration back end) of the Directory Server. All Global Administrative Group members have the same set of privileges and have the authority to send the administrative control.

2.3 Anticipating growth

After a directory has been deployed successfully, it is likely that the demands on it will increase over time. The number and types of users who are registered in the directory might increase, more applications might be written or converted to use the directory, and the number of people using the directory grows.

If you plan for this growth when you design the directory, it will be much easier to increase its capacity in the future.

2.3.1 Data considerations

The disk usage of the directory database increases with the number and size of entries. If you monitor the disk usage as you add entries, you can estimate the cost per entry so that you can estimate disk space requirements as your directory population grows. Then you can anticipate when you will need to make more storage available. If you add disks to the server, you can expand the main LDAP table space to include space on those disks. When you add containers to a DMS table space, the existing data might be rebalanced across all table spaces. This might take time, so it is best to plan this action during an adequate maintenance window.

2.3.2 Workload considerations

As the workload of the directory increases, there are two ways of increasing capacity:

- ▶ Faster servers (vertical scaling)
- ▶ More servers (horizontal scaling)

In most situations, read actions dominate the workload. Planning for horizontal scaling through replication allows you to exceed the capacity of the fastest single server and enables you to use less expensive servers. You can use replication to spread your directory across multiple servers if your applications can take advantage of them. If writes are a large part of the workload, this does not work as well, because updates must be replicated to all of the servers, so every server must process every update. In this case, you would need to partition the directory so different parts are on different servers, thus dividing the write workload. The proxy server can be used to make the partitioned servers still look like a single directory.

If you can design your directory tree so that different applications use different parts of it, it is much easier to partition the directory, and it might not be necessary to use a proxy server. As your directory workload grows, you would be able to move the distinct parts of the directory tree, and their associated applications, off to separate servers. This can allow for a much easier scaling path.



Time to do a health check

The Directory Server is a constantly changing entity. From time to time, a health check must 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 must be answered to make concrete recommendations. You must know what you have to work with before you can make the right decisions.

3.1 Questions to ask

Here is a list of questions to ask about each Lightweight Directory Access Protocol (LDAP) environment:

- ▶ What version of Directory Server is it?
- ▶ What fix packs were 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 were applied to IBM DB2 database?
- ▶ Is SSL being used in the environment?
- ▶ What version of IBM Global Security Kit (GSKit) is being used?
- ▶ What operating system (OS) version is the Directory Server running on?
- ▶ What fix packs were 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)?
- ▶ What is the current replication topology?
- ▶ Do you have a wide area network (WAN) environment? If you do, how many WANs and how fast are they?
- ▶ What is your failover process?
- ▶ What is your backup process?
- ▶ Do you have distributed or central administration?

3.2 Diagrams and layouts of the system

In addition to asking these questions, get the following information:

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

3.3 Configurations, logs, and outputs

In determining the health of the directory, review the logs first. The following section describes the log files that are available to you as part of the review process.

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

The following list shows where the log and configuration files are for a specified instance. (The <Location> and <Name> values can be found by using the `idsi list -a` command. These values are specified by the user during the instance configuration.)

- ▶ Version 6.x:
 - Directory Server configuration files:
 - UNIX: <Location>/idsslapd-<Name>/etc/ibmslapd.conf
 - Windows: <Location>\idsslapd-<Name>\etc\ibmslapd.conf
 - Directory Server error logs:
 - UNIX: <Location>/idsslapd-<Name>/logs
 - Windows: <Location>\idsslapd-<Name>\logs
 - 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 paper includes tools that assist you in problem determination and provide potential solutions.

- ▶ For Directory Server Version 6.x and later:
 - `ibmslapd.conf`: This is where you look for server configuration information such as LDAP cache settings and DB2 database connections.
 - `ibmslapd.log`: This is where you look for informational, warning, and error messages to identify potential problems with the schema, replication, or other server operational issues.
 - `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 the workload of the environment. It is also useful in identifying performance issues are covered later in this paper.
 - `lostandfound.log`: This log shows replication conflicts, if any. If there are many conflicts being logged, that affects the performance of the server.
- ▶ For IBM DB2 Version 9.x and later:
 - `db2diag.log`: This log tells you about any problems with the database and shows when someone initiates 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.

3.4 Scripts to help gather information

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

Note: See Appendix G, “Additional material” on page 239, for how to get the additional files that are mentioned throughout this IBM Redpaper publication.

3.4.1 perfcheck_database.sh

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

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

It also prints **db2look** output that shows the structure of the database instance, along with how each of the tables and indexes is built. To use the script, follow these steps:

1. Run the **sudo** (or **su**) command as the DB2 instance owner. If the user and instance were both set up correctly, the DB2 environment is sourced automatically for the instance owner at login (by adding lines to your user's login rc files that source the db2profile file). If this does not happen, you can source it manually, following the example. The db2profile script file is in the sqllib subdirectory under the instance owner's home directory. Then, source the script with the dot (.) operator. If, for example, your instance owner is ldapdb2, this is the command:

```
#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
```

Where the lone argument is the name of the Directory Server database.

3. When you are finished, type exit to return to the root shell.

3.4.2 perfcheck_runstats.sh

This gathers information about when **runstats** was last run on the DB2 tables and the indexes that are used in the LDAP instance, along with information about the tables and the distribution of data that influences the DB2 optimizer. This is run as the DB2 instance owner.

The script collects data about each of the tables and indexes, which list the date and time that **runstats** was run. It also collects the statistics about the tables in the database.

1. Run the **sudo** (or **su**) command as the DB2 instance owner, and run the script. If, for example, your instance owner is ldapdb2, this is the command:

```
#su - ldapdb2
```

2. Change to your ldapscripts directory:

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

3. When you are finished, type exit to return to root ID.

3.4.3 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 OS:

- ▶ Memory
- ▶ SWAP space
- ▶ Processor information
- ▶ Kernel bit width
- ▶ Outputs for:
 - vmstat
 - iostat
- ▶ Oracle Solaris and IBM AIX® systems only
 - prstat output
- ▶ Kernel parameters

For example:

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

3.4.4 perfcheck_ldap.sh

This script has been rewritten, because the original version could be used only against a server that allowed anonymous binds, ran on the default port, and allowed non-SSL connections.

This script gathers the following information:

- ▶ Dumps that contain the entire contents of the config file.
- ▶ Outputs for the following **idsldapsearch** commands:
 - **RootDSE search:**

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 that is being used. Depending on the version of LDAP, it also shows you whether you are in configuration mode or not.
 - **cn=monitor search:**

Depending on the version of LDAP, this search gives you a snapshot of the directory statistics from the last time that it was started to the point of the snapshot. These stats get reset when you restart LDAP.
 - **cn=connections, cn=monitor search output:**

This search shows clients that were connected when the search was run.

For example:

```
# ./perfcheck_ldap.sh -D cn=root -w secret -Z /opt/ibm/ldap/certs/server.kb -P secret -p 636 -o /tmp/perfcheck_ldap.out
```

Calling the script with the **-h** flag returns a list of options. The options in this script were created to be as close as possible to the options for the **idsldapsearch** command.

3.5 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 that are running against the database.

Collecting system monitor data introduces processing overhead for the database manager. For example, to calculate the execution time of SQL statements, the database manager must make calls to the operating system to get time stamps before and after the execution of every statement. These types of system calls are usually expensive. Another form of overhead that is incurred by the system monitor is increased memory use. For every monitor element tracked by the system monitor, the database manager memory stores the collected data.

Care must be taken with turning on the **DFT_MON_TABLE** monitor switch, because this results in a performance hit on the database. Enable this switch only when it is needed, and then disable it.

The monitors shown in Table 3-1 can be turned on to gather stats. By default, all switches are turned off, except DFT_MON_TIMESTAMP.

Table 3-1 DB2 monitor overview

Monitor switch	DBM parameter	Information provided
BUFFERPOOL	DFT_MON_BUFPOOL	Number of reads and writes, time taken of all 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 and stop time, statement identification
TABLE	DFT_MON_TABLE	Measure of activity (rows read and written)
UOW	DFT_MON_UOW	Start and end times, completion status of unit of work
TIMESTAMP	DFT_MON_TIMESTAMP	Time stamps

We have provided scripts to help you turn on and turn off these monitor switches. Each time that you turn these monitor switches on or off, you must 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 do not work). The `db2profile` script file is in the `sqllib` subdirectory under the instance owner's home directory. If you must tailor this file, follow the comments inside of 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 your `ldapscripts` directory. Run one of the three monitor scripts that follows 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. Restart the Directory Server:

```
idsslapd -I ldapdb2
```

3.6 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 the 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 entry cache often does not increase performance and, in fact, can degrade performance due to entry cache invalidation. In these instances, it is better to allocate the memory to DB2 and bypass the Directory Server caches. We suggest that you decrease the entry cache setting to 100 and use the performance tools included in this paper to determine what the other LDAP cache settings need to be.

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

The important thing to remember when you are tuning the Directory Server is that it is a continual process. The interaction between the transaction mix, the Directory Server process, and DB2 must be evaluated and tuned regularly 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 to compare. The benchmarks help you identify when your system is not running optimally. When identified, further testing is required to improve the outcome.



Tips for tuning DB2 performance

In this chapter, we explain how the following tools can help you tune your DB2 environment:

- ▶ Tuning DB2 and Directory Server caches and parameters
- ▶ Database maintenance utility: `idsdbmaint`
- ▶ DB2 self-tuning memory
- ▶ More tuning resources

4.1 Tuning DB2 and Directory Server caches and parameters

The **idsperftune** tool can monitor and tune parameters to achieve optimal performance for the Directory Server and the associated DB2 database. Along with tuning parameters, this tool includes a tuning abstraction that provides an interface within the Directory Server to tune the database. The key areas of the **idsperftune** tool are the directory cache, DB2 buffer pools, and various DB2 parameters. The LDAP caches that can be tuned by the tool are the Entry cache, Filter cache, and group member cache.

Configuration files: The **idsperftune** tool uses a configuration file in the `<user_home>/etc/idsperftune_input.conf` directory.

It also records statistics in the `<user_home>/logs/perftune_stats.log`.

4.1.1 Determine the number of entries and average entry size with **idsperftune**

At the command prompt, you can use the **idsperftune -I <instance_name> -s** option to determine the number of entries and to identify the average entry size in a Directory Server, as demonstrated in Example 4-1. When it is run with the **-s** parameter, the tool fetches the number of entries and average entry size information and records the information in the `perftune_input.conf` file.

Example 4-1 Determining number of entries and identifying average entry size in a Directory Server

```
$ ./idsperftune -I idslldap -s
GLPWRP123I The program '/opt/ibm/ldap/V6.3/sbin/32/perftune' is used with the
following arguments '-I idslldap -s'.
GLPCTL113I Largest core file size creation limit for the process (in bytes):
'0'(Soft limit) and '-1'(Hard limit).
GLPCTL119I Maximum Data Segment(Kbytes) soft ulimit for the process is -1 and the
prescribed minimum is 262144.
GLPCTL119I Maximum File Size(512-bytes block) soft ulimit for the process is -1
and the prescribed minimum is 2097152.
GLPCTL122I Maximum Open Files soft ulimit for the process is 1024 and the
prescribed minimum is 500.
GLPCTL121I Maximum Stack Size(Kbytes) soft ulimit for the process was 8192 and it
is modified to the prescribed minimum 10240.
GLPCTL119I Maximum Virtual Memory(Kbytes) soft ulimit for the process is -1 and
the prescribed minimum is 1048576.
GLPSRV200I Initializing primary database and its connections.
GLPPFT009I Fetching the total number of entries and the average size of an entry
in the directory.
GLPPFT024I Updated the configuration file
/home/idslldap/idsslldap-idslldap/etc/perftune_input.conf.
GLPPFT030I The command completed successfully.
```

This sets the values for the following parameters in the `idsperftune_input.conf` file

- ▶ `TDS_TOTAL_ENTRY=100062`
- ▶ `TDS_AVG_ENTRY_SZ=671`

Similarly, you can also use the Directory Server GUI tool, **idsxcfg** (Figure 4-1 on page 21) to determine the number of entries and average entry size. The “Performance Tuning” section of the **idsxcfg** utility invokes **idsperftune** tool. The option to “load from server instance database”

executes `idsperftune` tool with the `-s` option to populate the number of entries and average entry size.

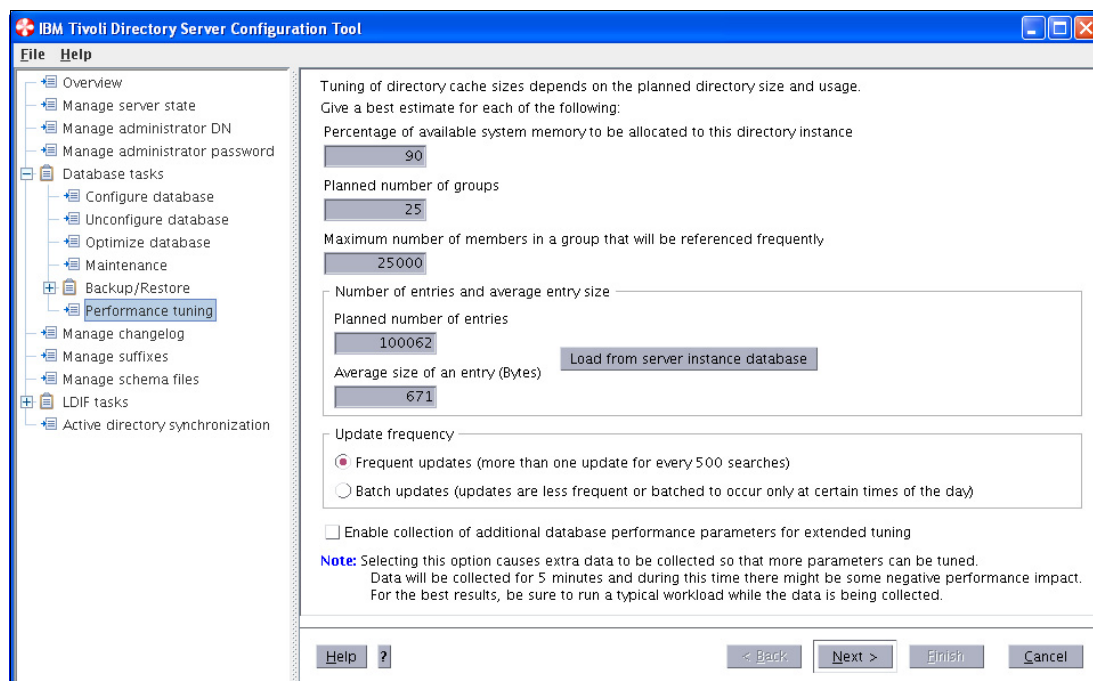


Figure 4-1 Performance tuning through the Directory Server Configuration tool

4.1.2 Basic tuning

Based on the number of entries, the average entry size, and the available system memory, `idsperftune` sets the values for the Directory Server cache and DB2 buffer pools.

From the console, you can run `idsperftune` with the `-B` option to tune the caches, as shown in Example 4-2.

Example 4-2 Basic tuning

```
# ./idsperftune -I idsperftune -B
GLPWRP123I The program '/opt/ibm/ldap/V6.3/sbin/32/perftune' is used with the
following arguments '-I idsperftune -B'.
GLPCTL113I Largest core file size creation limit for the process (in bytes):
'O'(Soft limit) and '-1'(Hard limit).
GLPCTL119I Maximum Data Segment(Kbytes) soft ulimit for the process is -1 and the
prescribed minimum is 262144.
GLPCTL119I Maximum file size (512-bytes block) soft ulimit for the process is -1
and the prescribed minimum is 2097152.
GLPCTL122I Maximum Open Files soft ulimit for the process is 1024 and the
prescribed minimum is 500.
GLPCTL121I Maximum Stack Size(Kbytes) soft ulimit for the process was 8192 and it
is modified to the prescribed minimum 10240.
GLPCTL119I Maximum Virtual Memory(Kbytes) soft ulimit for the process is -1 and
the prescribed minimum is 1048576.
GLPSRV200I Initializing primary database and its connections.
GLPPFT010I Performing basic tuning operation.
GLPPFT003I Parsing the input configuration file
/home/sujay1/idsslapd-sujay1/etc/perftune_input.conf.
```

```
GLPPFT017I Updated the status file
/home/idsldap/idsslapd-idsldap/logs/perftune_stat.log.
GLPPFT030I The command completed successfully.
```

This sets the values for the following parameters in the `perftune_stats.log` file. Unless **idsperftune** is run with the **-u** (update) option, none of the parameters are changed.

- ▶ TDS_ENTRY_CACHE=80049
- ▶ TDS_FILTER_CACHE=0
- ▶ TDS_GROUP_CACHE=25
- ▶ TDS_GROUP_MEMBER=25000
- ▶ IBMDEFAULTBP=AUTOMATIC
- ▶ LDAPBP=AUTOMATIC
- ▶ SYSTEM_MEMORY=154853.98
- ▶ SYS_MEM_AVL=TRUE

By default, **idsperftune** sets the entry cache to 80% of the number of entries in the Directory Server. It calculates the available system memory and sets the **SYS_MEM_AVL** parameter to **TRUE** if the system has sufficient memory to cache 80% of the entries. The tool sets the DB2 buffer pools (IBMDEFAULTBP, LDAPBP) to **AUTOMATIC** so that the DB2 self-tuning memory manager can choose the best memory allocation, based on its computations. The default percentage can be changed by the **-E** parameter.

Applying the basic tuning with the update parameter

The **idsperftune** tool can update the Directory Server cache and DB2 buffer pool parameters when run with the **-u** (update) options. The tool checks whether there is sufficient system memory available before it updates any Directory Server or DB2 parameters.

Running the idsperftune tool through the user interface (idsxcfg)

The Directory Server user interface the `idsxcfg` Performance Tuning section can be used to start the **idsperftune** tool. From that section, select the next panel to display the Basic and Advance attributes (see Figure 4-2 on page 23).

Tip: The user interface allows you to combine basic and advanced tuning into a single dialog window. After you have reviewed the performance parameters, you can update the parameters to Directory Server and DB2 parameters by selecting the **Update** option.

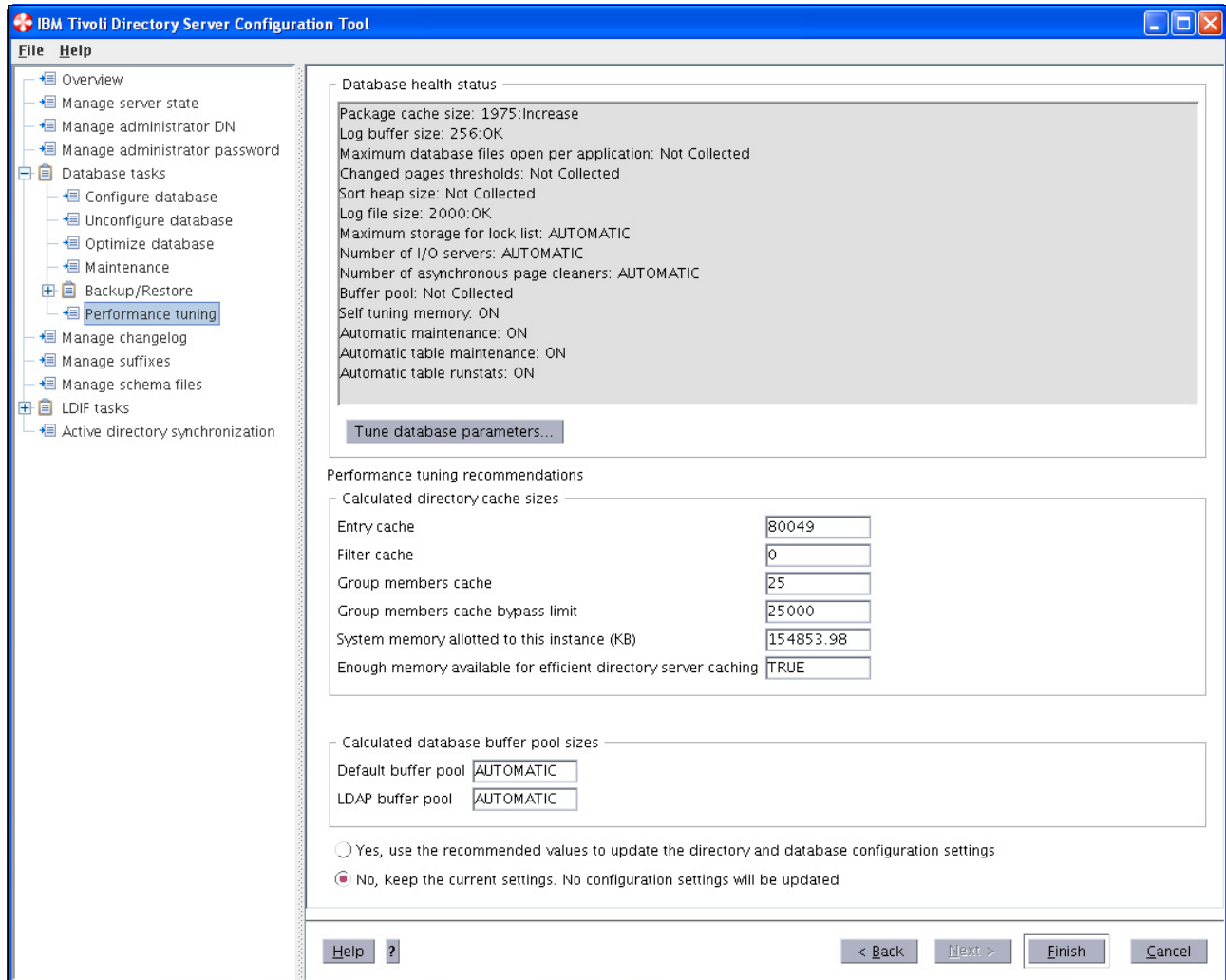


Figure 4-2 Performance tuning through the Directory Server configuration tool, second panel

Clicking **Tune database parameters** starts the **idsperf tune** tool in Advanced tuning mode.

4.1.3 Advanced tuning

In Advanced tuning mode, you can use **idsperf tune** to manipulate advanced parameters that are specifically related to DB2.

When you use the command-line interface, advanced tuning can be started by using the **-A** option, as shown in Example 4-3.

Example 4-3 Advanced tuning options

```
$ ./idsperf tune -I idslldap -A
GLPWRP123I The program '/opt/ibm/ldap/V6.3/sbin/32/perftune' is used with the
following arguments '-I idslldap -A'.
GLPCTL113I Largest core file size creation limit for the process (in bytes):
'0'(Soft limit) and '-1'(Hard limit).
GLPCTL119I Maximum Data Segment(Kbytes) soft ulimit for the process is -1 and the
prescribed minimum is 262144.
```

GLPCTL119I Maximum File Size (512 bytes block) soft ulimit for the process is -1 and the prescribed minimum is 2097152.
 GLPCTL122I Maximum Open Files soft ulimit for the process is 1024 and the prescribed minimum is 500.
 GLPCTL121I Maximum Stack Size(Kbytes) soft ulimit for the process was 8192 and it is modified to the prescribed minimum 10240.
 GLPCTL119I Maximum Virtual Memory(Kbytes) soft ulimit for the process is -1 and the prescribed minimum is 1048576.
 GLPSRV200I Initializing primary database and its connections.
 GLPPFT011I Performing advanced tuning operation.
 GLPPFT016I Successfully backed up DB2 parameter values in the perftune_stat.log file.
 GLPPFT013I Performing DB2 health check operation.
 GLPPFT017I Updated the status file
 /home/idsldap/idsslapd-idsldap/logs/perftune_stat.log.
 GLPPFT030I The command completed successfully.

In Advanced tuning mode, the **idsperftune** tool verifies the tuning parameters and records recommendations in the perftune_stat.log file in the following format:

<DB2 parameters>=<Current Value>:<Recommendation>

The result can be any of these responses or recommendations:

- ▶ <Not Collected>
- ▶ <OK>
- ▶ <Increase>
- ▶ <Decrease>

You can increase or decrease the parameter values by using the **set database parameters** option.

You can run **idsperftune** in Advanced mode through the Directory Server configuration tool user interface (idsxcfg) shown previously in Figure 4-2 on page 23. After selecting **Tune database parameters**, you see the dialog in Figure 4-3.

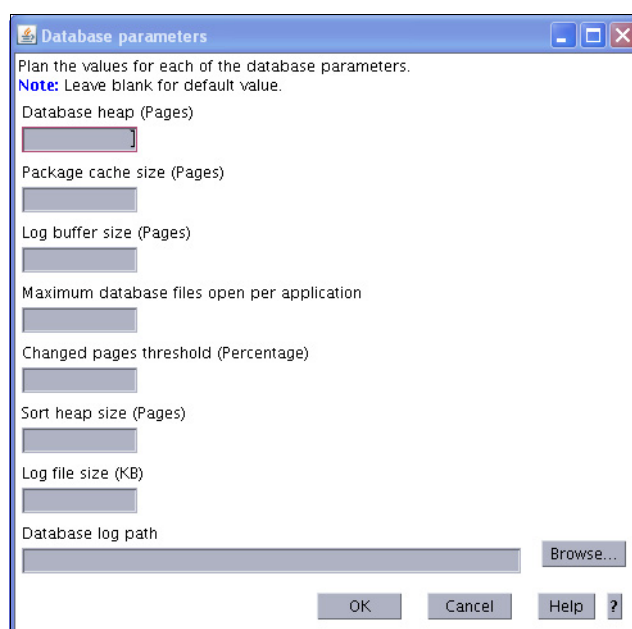


Figure 4-3 Advanced configuration for database parameters

After the parameters are tuned, the changes can be propagated to the Directory Server and DB2 by using the **-u** (update) option as shown in Example 4-4.

Example 4-4 Update the advanced tuning parameters

```
$ ./idsperftune -I idsldap -B -A -u
GLPWRP123I The program '/opt/ibm/ldap/V6.3/sbin/32/perftune' is used with the
following arguments '-I sujay1 -B -A -u'.
GLPCTL113I Largest core file size creation limit for the process (in bytes):
'O'(Soft limit) and '-1'(Hard limit).
GLPCTL119I Maximum Data Segment(Kbytes) soft ulimit for the process is -1 and the
prescribed minimum is 262144.
GLPCTL119I Maximum File Size (512 bytes block) soft ulimit for the process is -1
and the prescribed minimum is 2097152.
GLPCTL122I Maximum Open Files soft ulimit for the process is 1024 and the
prescribed minimum is 500.
GLPCTL121I Maximum Stack Size(Kbytes) soft ulimit for the process was 8192 and it
is modified to the prescribed minimum 10240.
GLPCTL119I Maximum Virtual Memory(Kbytes) soft ulimit for the process is -1 and
the prescribed minimum is 1048576.
GLPSRV200I Initializing primary database and its connections.
GLPPFT010I Performing basic tuning operation.
GLPPFT003I Parsing the input configuration file
/home/sujay1/idsslapd-sujay1/etc/perftune_input.conf.
GLPPFT017I Updated the status file
/home/idsldap/idsslapd-idsldap/logs/perftune_stat.log.
GLPPFT011I Performing advanced tuning operation.
GLPPFT016I Successfully backed up DB2 parameter values in the perftune_stat.log
file.
GLPPFT013I Performing DB2 health check operation.
GLPPFT005I Successfully backed up the ibmslapd.conf file to
/home/sujay1/idsslapd-sujay1/logs/ibmslapd.log.save.
GLPPFT014I Updated directory cache and DB2 BUFFERPOOL.
GLPPFT003I Parsing the input configuration file
/home/sujay1/idsslapd-sujay1/etc/perftune_input.conf.
GLPPFT017I Updated the status file
/home/sujay1/idsslapd-sujay1/logs/perftune_stat.log.
GLPPFT030I The command completed successfully.
```

Turning DB2 monitoring on and off

You can use the **idsperftune** tool to turn the DB2 monitoring option on and off. Turning on the DB2 monitor switch enables DB2 and the **idsperftune** tool to gather more information that can potentially lead to better tuning of the performance parameters. It is best to turn on the monitoring switch while the server is running and to perform regular Directory Server operations to collect better statistics for performance tuning. To turn on the DB2 monitor switch for DB2 buffer pools and sort, issue **idsperftune** with the **-m** flag (and, optionally, **-A** flag, which also captures a DB2 snapshot after 5 minutes). To disable the DB2 monitoring issue **idsperftune** with the **-o** flag.

4.2 Database maintenance utility: idsdbmaint

Directory Server provides a utility to perform DB2 database maintenance such as index reorganization, row compression, and table space conversion.

4.2.1 Index reorganization

Over time, multiple updates to the database can fragment data and indexes, so reorganize DB2 indexes as part of regular maintenance. Use the Directory Server database maintenance utility, **idsdbmaint**, for a DB2 database that is associated with the Directory Server instance. It provides an option to reorganize indexes on the tables within the defined schema. You can also use it to compress rows in tables and to convert DB2 table space.

Example 4-5 Using the idsdbmaint utility

```
$ ./idsdbmaint -I idsldap -i
GLPWRP123I The program '/opt/ibm/ldap/V6.3/sbin/32/dbmaint' is used with the
following arguments '-I idsldap -i'.
GLPCTL113I Largest core file size creation limit for the process (in bytes):
'O'(Soft limit) and '-1'(Hard limit).
GLPCTL119I Maximum Data Segment(Kbytes) soft ulimit for the process is -1 and the
prescribed minimum is 262144.
GLPCTL119I Maximum File Size (512 bytes block) soft ulimit for the process is -1
and the prescribed minimum is 2097152.
GLPCTL122I Maximum Open Files soft ulimit for the process is 1024 and the
prescribed minimum is 500.
GLPCTL121I Maximum Stack Size(Kbytes) soft ulimit for the process was 8192 and it
is modified to the prescribed minimum 10240.
GLPCTL119I Maximum Virtual Memory(Kbytes) soft ulimit for the process is -1 and
the prescribed minimum is 1048576.
GLPSRV200I Initializing primary database and its connections.
GLPDBA036I Index reorganization task will be performed.
GLPDBA021I All Index on table 'ACLINHERIT' will be reorganized.
GLPDBA044I The table 'ACLINHERIT' has been reorganized.
GLPDBA046I All statistics on table 'IDSLDAP.ACLINHERIT' have been updated.
GLPDBA021I All Index on table 'ACLPERM' will be reorganized.
GLPDBA044I The table 'ACLPERM' has been reorganized.
GLPDBA046I All statistics on table 'IDSLDAP.ACLPERM' have been updated.
GLPDBA021I All Index on table 'ACLPROP' will be reorganized.
...
GLPDBA046I All statistics on table 'IDSLDAP.TELETEXTERMINALID' have been updated.
GLPDBA021I All Index on table 'TELEXNUMBER' will be reorganized.
GLPDBA044I The table 'TELEXNUMBER' has been reorganized.
GLPDBA046I All statistics on table 'IDSLDAP.TELEXNUMBER' have been updated.
GLPDBA021I All Index on table 'TITLE' will be reorganized.
GLPDBA044I The table 'TITLE' has been reorganized.
GLPDBA046I All statistics on table 'IDSLDAP.TITLE' have been updated.
GLPDBA021I All Index on table 'UID' will be reorganized.
GLPDBA044I The table 'UID' has been reorganized.
GLPDBA046I All statistics on table 'IDSLDAP.UID' have been updated.
GLPDBA021I All Index on table 'UNIQUEMEMBER' will be reorganized.
GLPDBA044I The table 'UNIQUEMEMBER' has been reorganized.
GLPDBA046I All statistics on table 'IDSLDAP.UNIQUEMEMBER' have been updated.
GLPDBA021I All Index on table 'USERPASSWORD' will be reorganized.
GLPDBA044I The table 'USERPASSWORD' has been reorganized.
GLPDBA046I All statistics on table 'IDSLDAP.USERPASSWORD' have been updated.
GLPDBA021I All Index on table 'X121ADDRESS' will be reorganized.
GLPDBA044I The table 'X121ADDRESS' has been reorganized.
GLPDBA046I All statistics on table 'IDSLDAP.X121ADDRESS' have been updated.
GLPDBA027I Index reorganization task is complete.
```

4.2.2 Row compression

DB2 row compression uses a static dictionary-based compression algorithm to compress data by row. This allows repeating patterns that span multiple column values within a row to be replaced with shorter symbol strings. Table rows are compressed by reconstructing the compression dictionary and compressing the information to eliminate fragmented data. Data compression reduces the space that is required for the directory, reduces I/O, and improves performance. The **idsdbmaint** tool compresses the row in the following way:

1. Queries DB2 `syscat.tables` and fetches all the tables that belong to the Directory Server instance.
 2. Inspects the table and fetches the row compression estimates for each table.
 3. If the compression estimate is more than 30%, the tool takes these actions:
 - Alters the table to enable **ROW COMPRESSION**.
 - Runs the **DB2 REORG** command on the table and builds a new compression dictionary.
 - After running **DB2 REORG** on the table, all of the statistics on the table are updated by running **DB2 RUNSTATS** on the table.
- Creates a new compression dictionary.

You can optimize the database by running the **idsdbmaint** command with the row compression option from the command prompt. For example:

```
idsdbmaint -I instance_name -r
```

The **idsdbmaint** tool can inspect tables for any compression benefit. If the compression benefit is greater than 30%, the tool selects the table for compression, as shown in Example 4-6.

Example 4-6 Running the idsdbmaint tool for compression

```
# ./idsdbmaint -I sujay1 -r
GLPWRP123I The program '/opt/ibm/ldap/V6.3/sbin/32/dbmaint' is used with the
following arguments '-I sujay1 -r'.
GLPCTL113I Largest core file size creation limit for the process (in bytes):
'0'(Soft limit) and '-1'(Hard limit).
GLPCTL119I Maximum Data Segment(Kbytes) soft ulimit for the process is -1 and the
prescribed minimum is 262144.
GLPCTL119I Maximum File Size (512 bytes block) soft ulimit for the process is -1
and the prescribed minimum is 2097152.
GLPCTL122I Maximum Open Files soft ulimit for the process is 1024 and the
prescribed minimum is 500.
GLPCTL121I Maximum Stack Size(Kbytes) soft ulimit for the process was 8192 and it
is modified to the prescribed minimum 10240.
GLPCTL119I Maximum Virtual Memory(Kbytes) soft ulimit for the process is -1 and
the prescribed minimum is 1048576.
GLPSRV200I Initializing primary database and its connections.
GLPDBA037I Row compression task will be performed.
GLPDBA023I Index Reorganization or Row Compression might not provide performance
gain for the table 'ACLINHERIT'.
GLPDBA043I The table 'ACLPERM' has a compression benefit of '59%'.
GLPDBA019I The table 'ACLPERM' will be compressed.
GLPDBA034I Row compression has been enabled for the table 'ACLPERM'.
GLPDBA044I The table 'ACLPERM' has been reorganized.
GLPDBA046I All statistics on table 'SUJAY1.ACLPERM' have been updated.
...
```

GLPDBA043I The table 'TELEPHONENUMBER' has a compression benefit of '40'%.

GLPDBA019I The table 'TELEPHONENUMBER' will be compressed.

GLPDBA034I Row compression has been enabled for the table 'TELEPHONENUMBER'.

GLPDBA044I The table 'TELEPHONENUMBER' has been reorganized.

GLPDBA046I All statistics on table 'SUJAY1.TELEPHONENUMBER' have been updated.

GLPDBA023I Index Reorganization or Row Compression might not provide performance gain for the table 'TELETEXTERMINALID'.

GLPDBA043I The table 'TELEXNUMBER' has a compression benefit of '41'%.

GLPDBA019I The table 'TELEXNUMBER' will be compressed.

GLPDBA034I Row compression has been enabled for the table 'TELEXNUMBER'.

GLPDBA044I The table 'TELEXNUMBER' has been reorganized.

GLPDBA046I All statistics on table 'SUJAY1.TELEXNUMBER' have been updated.

GLPDBA043I The table 'TITLE' has a compression benefit of '38'%.

GLPDBA019I The table 'TITLE' will be compressed.

GLPDBA034I Row compression has been enabled for the table 'TITLE'.

GLPDBA044I The table 'TITLE' has been reorganized.

GLPDBA046I All statistics on table 'SUJAY1.TITLE' have been updated.

GLPDBA043I The table 'UID' has a compression benefit of '51'%.

GLPDBA019I The table 'UID' will be compressed.

GLPDBA034I Row compression has been enabled for the table 'UID'.

GLPDBA044I The table 'UID' has been reorganized.

GLPDBA046I All statistics on table 'SUJAY1.UID' have been updated.

GLPDBA024I The available data is not enough to perform inspection for the table 'UNIQUEMEMBER'.

GLPDBA024I The available data is not enough to perform inspection for the table 'USERPASSWORD'.

GLPDBA023I Index Reorganization or Row Compression might not provide performance gain for the table 'X121ADDRESS'.

GLPDBA028I Row compression task is complete.

4.2.3 Table space conversion

The **idsdbmaint** tool converts the table space type from SMS to DMS and from DMS to SMS. The LDAPSPACE and USERSPACE1 table spaces are considered by the **idsdbmaint** tool. When the **idsdbmaint** command is run with **-t <ts_type>**, it converts a table space type from SMS to DMS and from DMS to SMS, respectively, depending on the table space type that exists and the table space type to be converted. The valid values for the table space type are SMS and DMS. The **idsdbmaint** tool calculates the database size and determines the required disk space for the import and export of the user's data. If the required disk space is not available, the tool displays an appropriate error message and exits.

The **idsdbmaint** tool converts the table space from SMS to DMS in the following way:

1. Exports all of the data from the LDAPSPACE and USERSPACE1 table spaces, along with the table definitions.
2. Drops the SMS table space type.
3. Creates the DMS table space type.
4. Uses a REGULAR table space that uses a FILE container because it can automatically resize.
5. Reconstructs all of the tables within the table space and loads all of the data into the database.

The **idsdbmaint** tool converts the table space from DMS to SMS in the following way:

1. Exports all of the data from the LDAPSPACE and USERSPACE1 table spaces, along with the table definitions.
2. Drops the DMS table space type.
3. Creates the SMS table space type.
4. Uses a REGULAR table space that uses a PATH (directory) container because it can automatically resize.
5. Reconstructs all of the tables within the table space and loads all of the data into the database.

Examples:

Using the **idsdbmaint** command for table space conversion:

To convert an SMS table space to a DMS table space and to store the exported data in a mydata directory, run the following command:

```
idsdbmaint -I <instance_name> -t DMS -k  
           <instance_location>/<instance_name>/mydata
```

To specify a file container for LDAPSPACE table spaces while converting an SMS table space to a DMS table space and to store the exported data in a directory, run the following command:

```
idsdbmaint -I <instance_name> -t DMS -l  
           /disk/32K_ldapSPACE_container/ldapSPACE -k /disk/mydata
```

To specify a file container for USERSPACE1 table spaces while converting from an SMS table space to a DMS table space and to store the exported data in a directory, run the **idsdbmaint** command with the following arguments:

```
idsdbmaint -I <instance_name> -t DMS -u  
           /disk/container/userspace1 -k /disk/mydata
```

To specify a file container for LDAPSPACE and USERSPACE1 table spaces while converting from SMS to DMS and to store the exported data in a directory, run the following command:

```
idsdbmaint -I <instance_name> -t DMS  
           -l /disk/32K_ldapSPACE_container/ldapSPACE  
           -u /disk/container/userspace1 -k /disk/mydata
```

To convert a DMS table space to an SMS table space and to store the exported data in a directory called mydata, run the following command:

```
idsdbmaint -I <instance_name> -t SMS -k  
           <instance_location>/<instance_name>/mydata
```

To specify a container path for LDAPSPACE table spaces while converting from DMS to SMS and to store the exported data in a directory, run the following command:

```
idsdbmaint -I <instance_name> -t SMS  
           -l /disk/32K_ldapSPACE_container/ -k /disk/mydata
```

To specify a container path for USERSPACE1 table spaces while converting from DMS to SMS and to store the exported data in a directory, run the following command:

```
idsdbmaint -I <instance_name> -t SMS  
           -u /disk/userspace1_container/ -k /disk/mydata
```

To specify a file container for LDAPSPACE and USERSPACE1 table spaces while converting from DMS to SMS and to store the exported data in a directory, run the following command:

```
idsdbmaint -I <instance_name> -t SMS  
          -l /disk/32K_ldap_space_container/  
          -u /disk/userspace1_container/ -k /disk/mydata
```

Directory option: The directory that is specified with the **-k** option must have enabled read and write access for the DB2 instance owner.

4.3 DB2 self-tuning memory

Starting with IBM DB2 Version 9, a new memory-tuning feature simplifies the task of memory configuration by automatically setting values for several memory configuration parameters. When enabled, the memory tuner dynamically distributes available memory resources among several memory consumers, including sort, the package cache, the lock list, and buffer pools.

The following memory consumers can be enabled for self-tuning:

- ▶ Buffer pools (controlled by the **ALTER BUFFERPOOL** and **CREATE BUFFERPOOL** statements)
- ▶ Package cache (controlled by the **pckcachesz** configuration parameter)
- ▶ Locking memory (controlled by the **locklist** and **maxlocks** configuration parameters)
- ▶ Sort memory (controlled by the **sheapthres_shr** and the **sortheap** configuration parameters)
- ▶ Database shared memory (controlled by the **database_memory** configuration parameter)

More information: See the “Self-tuning memory” topic in the DB2 Information Center:

<http://bit.ly/ljsIgYa>

The **idsperftune** tool checks if self-tuning memory is enabled. If the **idsperftune** tool is run in Advanced mode by providing the **-A** and **-u** options, self-tuning memory is enabled if the tool detects that it is disabled.

To get advanced tuning recommendations with the monitor switches set to ON, run the **idsperftune** command with the following arguments:

```
idsperftune -I <instance_name> -A -m
```

The monitor switches are set to OFF after the tool completes the operation.

To update the database with the suggested DB2 parameters in advanced tuning with the monitor switches set to ON, run the command with the following arguments:

```
idsperftune -I <instance_name> -A -u -m
```

The monitor switches are set to OFF after the tool completes the operation.

4.4 More tuning resources

Performance tuning of the Directory Server environment is specific to the applications that interact with the Directory Server. For IBM Security Access Manager and IBM Security Identity Manager environments, review the application recommendations and resources that are available. Many of the scripts that are provided in the Security Identity Manager Performance Tuning Guide can be used to tune the Directory Server environment for optimal performance. (See the link that follows.)

As with any performance tuning activity, it is important to test and evaluate the changes that are implemented to ensure that the environment is performing as anticipated. Also, it is important to keep in mind that performance tuning is an ongoing and iterative process. As new workloads are introduced and as operations increase or new applications are added, it is important to test and measure the changes before going into the production environment.

IBM Security Identity Manager Performance Tuning scripts are available on this web page:

<http://www.ibm.com/support/docview.wss?uid=swg27013557>



DB2 settings related to LDAP

All of the settings that are covered in this chapter can be changed in the `db2_tunings.sh` script. By using the DB2-Config-calc-tool-template in the Microsoft Excel worksheet results tab, along with this section, you can get a better understanding of appropriate values for these settings:

- ▶ IBM DB2 self-tuning memory overview
- ▶ Adjusting the buffer pool and sort heap threshold settings
- ▶ Self-tuning memory and the `sortheap` parameter
- ▶ Self-tuning memory and the `pckcachesz` parameter
- ▶ Self-tuning memory and the `locklist` parameter
- ▶ Self-tuning memory and health checks
- ▶ DB2SET commands

5.1 IBM DB2 self-tuning memory overview

Starting in IBM DB2 Version 9, a memory-tuning feature called the *self-tuning memory manager (STMM)* can simplify the task of memory configuration by automatically setting values for several memory configuration parameters. When enabled, this memory tuner dynamically distributes available memory resources among the following memory users: buffer pools, locking memory, package cache, and sort memory.

These memory-related database configuration parameters can be automatically tuned:

database_memory	Database shared memory size
locklist	Maximum storage for lock list
maxlocks	Maximum percent of lock list before escalation
pckcachesz	Package cache size
sheapthres_shr	Sort heap threshold for shared sorts
sortheap	Sort heap size

The **database_memory** parameter determines the total amount of memory available for database-level memory areas. When it is set to **AUTOMATIC** the database memory grows or shrinks as needed, based on free operating system memory.

When a new directory instance is created, all of these parameters are set to **AUTOMATIC** by default. If the instance was migrated from a previous release, the previous database configuration remains intact. Therefore, it is important to review and update the configuration to take advantage of the DB2 STMM feature.

5.2 Adjusting the buffer pool and sort heap threshold settings

When the instance-level **sheapthres** is set to 0, the sort memory use is tracked at the database level only. Memory allocation for sorts is constrained by the value of the database-level **sheapthres_shr** configuration parameter.

Automatic tuning of **sheapthres_shr** is allowed only when the database manager **sheapthres** configuration parameter is set to 0.

5.3 Self-tuning memory and the sortheap parameter

This 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. If the sort is a *private* sort, this parameter affects agent private memory. If it is a *shared* sort, this parameter affects the database shared memory.

Each sort has a separate sort heap that is allocated by the database manager as needed. This sort heap is the area where data is sorted. If directed by the optimizer, a smaller sort heap than the one specified by this parameter is allocated by using information that is provided by the optimizer.

When this parameter is set to **AUTOMATIC**, self-tuning is enabled. The memory tuner dynamically sizes the memory area that is controlled by this parameter as the workload requirements change.

5.4 Self-tuning memory and the pckcachesz parameter

Caching packages allows the database manager to reduce its internal overhead by eliminating the need to access the system catalogs when reloading a package; or, in the case of dynamic SQL or XQuery statements, eliminating the need for compilation. Sections are kept in the package cache until one of the following occurs:

- ▶ The database is shut down.
- ▶ The package or dynamic SQL or XQuery statement is invalidated.
- ▶ The cache runs out of space.

This caching of the section for a static or dynamic SQL or XQuery statement can improve performance, especially when the same statement is used multiple times by applications that are connected to a database. This is especially important in a transaction-processing environment such as LDAP.

When this parameter is set to AUTOMATIC, it is enabled for self-tuning. When `self_tuning_mem` is set to ON, the memory tuner dynamically sizes the memory area that is controlled by `pckcachesz` as the workload requirements change.

5.5 Self-tuning memory and the locklist parameter

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 that are held by all applications that are concurrently connected to the database. Locking is the mechanism that the database manager uses to control concurrent access to data in the database by multiple applications. Both rows and tables can be locked. The database manager can also acquire locks for internal use. This is set to AUTOMATIC by default, which allows the memory tuner to dynamically size the memory area that is controlled by this parameter as the workload requirements change.

5.6 Self-tuning memory and health checks

If you upgraded to DB2 Version 9 from an earlier version of DB2 and you plan to use the self-tuning memory feature, configure the following health indicators to disable threshold or state checking:

Shared Sort Memory Utilization	<code>db.sort_shrmem_util</code>
Percentage of sorts that overflowed	<code>db.spilled_sorts</code>
Long Term Shared Sort Memory Utilization	<code>db.max_sort_shrmem_util</code>
Lock List Utilization	<code>db.locklist_util</code>
Lock Escalation Rate	<code>db.lock_escal_rate</code>
Package Cache Hit Ratio	<code>db.pkgcache_hitratio</code>

One of the objectives of the self-tuning memory feature is to avoid having memory that is allocated to a memory consumer when it is not immediately required. Therefore, use of the memory that is allocated to a memory consumer might approach 100% before more memory is allocated. By disabling these health indicators, you can avoid unnecessary alerts that are triggered by the high rate of memory use.

Instances that are created in DB2 Version 9 have these health indicators disabled by default.

5.7 DB2SET commands

Take a closer look at these three **DB2SET** commands:

- ▶ **DB2_PARALLEL_IO**
- ▶ **DB2_HASH_JOIN**
- ▶ **DB2_ANTIJOIN**

5.7.1 DB2_PARALLEL_IO

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

Procedure

Follow these steps for each table space that is stored on a RAID device:

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

And:

- A multiple of the **EXTENTSIZE**
4. Use the **DB2_PARALLEL_IO** registry variable to enable parallel I/O for the table space.

For LDAP, we strongly suggest that you set this variable to * any time that you are going to use RAID devices or put your database instance on storage area networks (SANs). For example:

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

5.7.2 DB2_HASH_JOIN

The **DB2_HASH_JOIN**, by default, is set to YES. For Lightweight Directory Access Protocol (LDAP), we suggest that you set this variable to NO. Our testing of a large directory produced better performance by turning off **HASH_JOIN** without producing any problems with the directory or LDAP commands with this set to NO, as in this example:

```
db2set DB2_HASH_JOIN=NO
```

5.7.3 DB2_ANTIJOIN

The **DB2_HASH_JOIN**, by default, is set to **NO**. When **YES** is specified, as in the following example, the optimizer searches for opportunities to transform **NOT EXISTS** subqueries into antijoins, which can be processed more efficiently by DB2. When **EXTEND** is specified, the optimizer searches for opportunities to transform both **NOT IN** and **NOT EXISTS** subqueries into antijoins.

```
db2set DB2_ANTIJOIN=YES
```



Using LDAP_MAXCARD and IBMSLAPD_USE_SELECTIVITY

This chapter explains table cardinality and how it can affect the IBM DB2 optimizer. It also provides information about the **LDAP_MAXCARD** and **LDAP_USE_SELECTIVITY** variable settings, what they are, and when to use them to influence the behavior of the DB2 search optimizer. We cover these topics:

- ▶ Table cardinality and performance
- ▶ LDAP_MAXCARD setting
- ▶ LDAP and DB2 Selectivity
- ▶ Additional indexes

6.1 Table cardinality and performance

In this section, we look at the LDAP_DESC and LDAP_ENTRY tables in more detail.

LDAP_DESC table

When DB2 processes 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 this 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 of the descendants of o=everything first to determine which of these are widgets or whether it is better to find all widgets first to determine which of these are descendants of o=everything. The actual query looks something like Example 6-1.

Example 6-1 LDAP query

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

Notice the use of the ? parameter marker. The actual value of the parameter marker is substituted in the query. When DB2 does the optimization, the optimizer does not know what values are to be substituted for the parameter markers. Therefore, the estimate is 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 is 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, so 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 expensive. It finds all of the descendants of o=everything first, and then it discards all of those that do not have objectclass=widgets. This results in poor 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 compared to 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 is not sufficiently selective. If the number of distinct values is high, DB2 concludes that the search is very selective. By setting the cardinality of the LDAP_DESC table artificially high, you can ensure that the descendant table is not queried first. There are multiple methods for setting the cardinality. This is one example (see 8.2, "Which runstats to use" on page 62 for more methods):

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

After this runs, the previous query for widgets objects in the o=everything tree becomes much faster. The difference might be on the order of 100 times, 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.

However, if the cardinality of the LDAP_DESC table was set as previously described, the DB2 optimizer chooses to find all person entries first instead and then to find which of those are descendants of ou=verySmall. This makes the search much slower. Again, the difference can be on the order of 100 times slower.

Therefore, the appropriate setting of cardinality for the LDAP_DESC table depends on whether most subtree searches are for small subtrees or large ones. This setting affects only subtree searches, not one-level or base searches.

For most LDAP usage patterns, searches of large subtrees predominate. Therefore, for most LDAP usage patterns, set the LDAP_DESC cardinality as previously described.

There are exceptions. Consider this example:

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

That search translates to this 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 this query 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.

LDAP_ENTRY table

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 this 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 very few of those are widgets. The corresponding SQL statement looks something like Example 6-2.

Example 6-2 SQL statement for the LDAP Entry table

```
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 child entries that the search base entry has because of the use of the ? placeholder. Therefore, DB2 assumes that there are few and finds all of the child entries first. Again, this is very expensive. To remedy this, you can set the cardinality of the entry table as very large:

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

6.1.1 Setting DB2 table statistics

The DB2 statistics tables are updated every time that **runstats** is used. When this happens, the artificial cardinality that was set in the table is overwritten. Therefore, whenever **runstats** is used, the cardinality of the LDAP_DESC must table be adjusted again (this is taken care of with the **tune_runstats.sh** script, which is covered in 8.2, “Which runstats to use” on page 62).

There are other ways to set and maintain DB2 table cardinality through settings in the IBM Directory Server.

6.2 LDAP_MAXCARD setting

IBM 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 an 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 to first find all descendants of the root of the search and then evaluate whether they match the search filter, or whether to 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 by using the *cardinality* for the LDAP_DESC table. 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:

Example 6-3 Search 1

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

Example 6-4 Search 2

```
ldapsearch -b “o=Acme, c=US” -s subtree objectclass=rareWidgets (type 2)
```

If **LDAP_MAXCARD** is off, the DB2 optimizer uses 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 always finds all descendants of the root of the search first, and then it checks to see which of these are of the required object class. This approach resolves Search 1 quickly because there are very few descendants of `ou=very small`. However, it works very poorly for Search 2. The `o=Acme` entry might have millions of descendants, and each must be checked to see whether it is the `rareWidgets` object class. Very few are.

If **LDAP_MAXCARD** is on, the DB2 optimizer never uses the LDAP_DESC table first because the cardinality is too large. Instead, it finds all entries that match the filter. This works very well for Search 2 because only a handful of entries are of the `rareWidgets` object class. 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

person as the object class, 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 average, most entries in the directory have few or no descendants. Therefore, given the default cardinality, the DB2 optimizer evaluates 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 significant effect. One value of the switch can make certain searches 10,000 times slower than the other value. If the setting is wrong, the server throughput is poor, with resource consumption running near 100%.

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

- ▶ In Version 5.2, it tunes the card by default unless the **LDAP_MAXCARD** environment variable is set to NO.
- ▶ In Version 5.2 Fix Pack 3, it does *not* tune the card by default unless the **LDAP_MAXCARD** environment variable is set to YES. If you are going to set it to YES, you must also update the **tune_runstats.sh** (this is covered in 8.2, “Which runstats to use” on page 62) and edit to either remark out the card line or unremark the `sysstat.tables set card` line for the **LDAP_DESC** or **LDAP_ENTRY** table.

In Versions 6.0 and 6.1, **MAX_CARD** works like this:

- ▶ If **LDAP_MAXCARD** is not set, the server sets **LDAP_DESC** card to 9E18 once, only when the server starts.
- ▶ If **LDAP_MAXCARD** is set to YES, the server sets it once every minute.
- ▶ If **LDAP_MAXCARD** is set to NO, the server does not set it at all.

For Versions 6.2 and 6.3

- ▶ If **LDAP_MAXCARD** is not set
- ▶ If **LDAP_MAXCARD** is set to YES, the server sets it once every minute.
- ▶ If **LDAP_MAXCARD** is set to NO, the server does not set it at all.

In practice, always set the value to either YES or NO.

6.3 LDAP and DB2 Selectivity

You can improve the performance of subtree searches on search bases that are high in the directory tree and have a large number of child searches by adding *Selectivity* to LDAP server-generated SQL statements. The inclusion of *Selectivity* in SQL statements influences the DB2 optimizer in the formation of access plans to resolve the search request by identifying which tables to access first during searches.

Determination of the entries that are high in the tree and have a large number of child queries is based on DB2 statistics (more about that follows). If a subtree is searched by using one of these entries as the search base, the *Selectivity* clause is added to the SQL query. This causes DB2 to use the search filter to narrow the search results before reading the table that identifies the entries that are descendants of a base in a search.

Consider these factors:

- ▶ The number of distinct values for a column = index cardinality
- ▶ The (number of rows) ÷ (cardinality) = Selectivity
- ▶ So, large number of rows ÷ small number of distinct values = high Selectivity (not good)

A lower Selectivity value tells DB2 to use indexes for that column. A higher Selectivity influences DB2 to not use indexes but, instead, to use the filter to reduce the possible result set first.

An IBM LDAP server determines which distinguished names (DNs) can benefit from using Selectivity this way:

If the number of descendants of a DN is > 1000 and the ratio of previous DN's number of descendants to this DN's number of descendants < 5000, DN is a candidate for using DB2 Selectivity. Selectivity is set to 0.9.

To use Selectivity, the DB2 registry for the database instance must have **DB2_SELECTIVITY** set to YES. This is done when you are creating a database instance or while migrating data from previous versions. It also requires setting one of these environment variables:

IBMSLDAPD_USE_SELECTIVITY = NO | YES

If it is not set or is set to NO, Selectivity is not used to influence the DB2 access plans.

If it is set to YES and **LDAP_MAXCARD** is not set to YES, Selectivity is used to influence DB2 actions for accessing data during large subtree searches.

6.4 Additional indexes

An additional index on the LDAP_DESC table is strongly advised. This index is especially important if **LDAP_MAXCARD** is set to ON. The index is not needed if you set **LDAP_MAXCARD OFF**. The name of the index is LDAP_DESC_DEID. (LDAP_DESC_AEID is always created.) You can check for its presence as Example 6-5 shows. (The commands that follow are based on the assumption that the DB2 instance name and LDAP database name are *LDAPDB2*, but substitute the correct values for your installation.)

Example 6-5 Check for LDAP_DESC_DEID

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

This select statement needs to show two indexes: LDAP_DESC_AEID and LDAP_DESC_DEID. If the second is not present, you can create it as Example 6-6 shows.

Example 6-6 Create LDAP_DESC_DEID

```
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
```



Tools and scripts

In this chapter, we introduce important tools and scripts, some of which are available to download from the ITSO website (see Appendix G, “Additional material” on page 239). We describe the following tools:

- ▶ 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

7.1 ITDSAUDIT.JAR

Understanding what the directory is doing and how well it is doing is necessary before, during, and after any tuning exercise. Also, 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 to be unable to demonstrate or measure the effects of tuning and the actual performance of the directory after tuning.

As with any client/server application, determining the source of performance problems can be difficult to diagnose. There might be server-side issues (poor performance to client requests), network throughput or instability issues, or it might be the client that is the (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, so their interaction must also be taken into account when you are evaluating the directories' performance (for example, if one application is using the most processor resources).

Along with the other tools described in this paper, the `itdsaudit.jar` file offers a look of the directory and the operations that are run against it by parsing and reporting the transactions as seen by the directory audit log. When the audit logging is enabled, the Directory Server provides a comprehensive list of client requests, parameters, response times, and success or failure results of the request for each transaction that is processed when the audit log is active.

7.1.1 Theory of operation

The audit log (a standard text file) provides a tremendous amount of information when it is enabled. The audit log can including the following information, depending on what level and options are selected:

- ▶ Directory control information (enabling audits, 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 log can be huge, based upon the transaction volume. So the difficulty with the log is extracting and turning the raw data into information that you can use. The `itdsaudit.jar` tool can assist with this.

7.1.2 Prerequisites

The `itdsaudit.jar` archive is a Java application that is packaged as an executable Java archive (JAR) file. It was developed for Java 1.5, but there is no known reason why it is not likely to run with any recent version of Java. These are the only prerequisites:

- ▶ Java must be available.
- ▶ The user must have Read permission for the input file.
- ▶ The user must be able to write to the input audit log location if you want PDF output.

7.1.3 Starting itdsaudit.jar application

The itdsaudit.jar file is simple to run, because it has no switches and expects only the file name of the input audit log on the command line. Use this command line:

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

Where:

java -jar itdsaudit.jar is the normal start command for the Java virtual machine (JVM) and tells the JVM to start the *main* entry point of the supplied JAR file name.

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

7.1.4 itdsaudit.jar error messages

There are three error conditions that can commonly occur when running itdsaudit.jar:

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

```
ITDS audit log parser version 0.2b
```

```
You must supply the file name to parse on the command line, for example: java  
-jar itdsaudit.jar <FILENAME>
```

```
itdsaudit.jar parses and reports (both to stdout and a PDF file) the absolute  
and statistical information that is contained within an Tivoli Directory Server  
audit log file. Note: the PDF file name is always <FILENAME>.pdf and any  
existing file with that name is 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 prints an error message and continues with standard output only:

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

7.1.5 itdsaudit.jar stdout output

The itdsaudit.jar file outputs to standard output (stdout) general statistical information that is found within the audit log. The format and output looks similar to Example 7-1 (depending upon the information that is contained within the audit log).

Example 7-1 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=abccompany,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 = erpolycymembership 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 =

```

7.1.6 itdsaudit.jar PDF output

The PDF generated from the `itdsaudit.jar` contains additional information and output in a format that is appropriate for inclusion in tuning before, during, and after documentation. The PDF file is broken down into the sections that follow.

Title page

The title page contains:

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

Summary

The Summary section contains multiple charts and tables for the information that was found.

- Transaction types and totals:

Table 7-1 provides a summary count of the transactions and their average times to complete, in milliseconds (ms). As this table shows, the average transaction time for searches 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 7-1 Transaction totals and types

Transaction types and totals		
Operation	Count	Average (ms)
Binds	6	1
Unbinds	6	0
Searches	41,406	183
Adds	0	0
Deletes	0	0
Modifies	914	35
Totals	42,332	180

Figure 7-1 shows this in graph format (it is the same information, but the graph shows the distribution by transaction types at a glance).

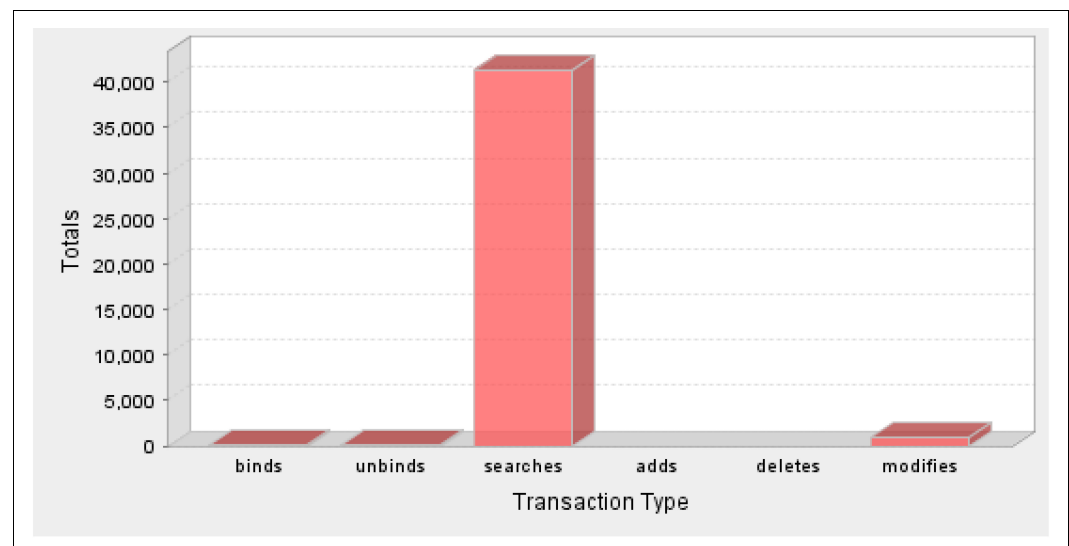


Figure 7-1 Transaction type

- Transaction types and totals without binds and unbinds:

Binds and unbinds can skew the average transaction rate because they are normally low-overhead operations. If the directory is serving as an authentication engine, they can hide issues with other transaction types. See Table 7-2 on page 48.

Table 7-2 Totals by type (without unbinds and binds)

Transaction types and totals		
Operation	Count	Average (ms)
Searches	41,406	183
Adds	0	0
Deletes	0	0
Modifies	914	35
Totals	42,332	180

This is shown in a graph format in Figure 7-2 (it is the same information, but the graph shows at a glance the distribution by transaction types).

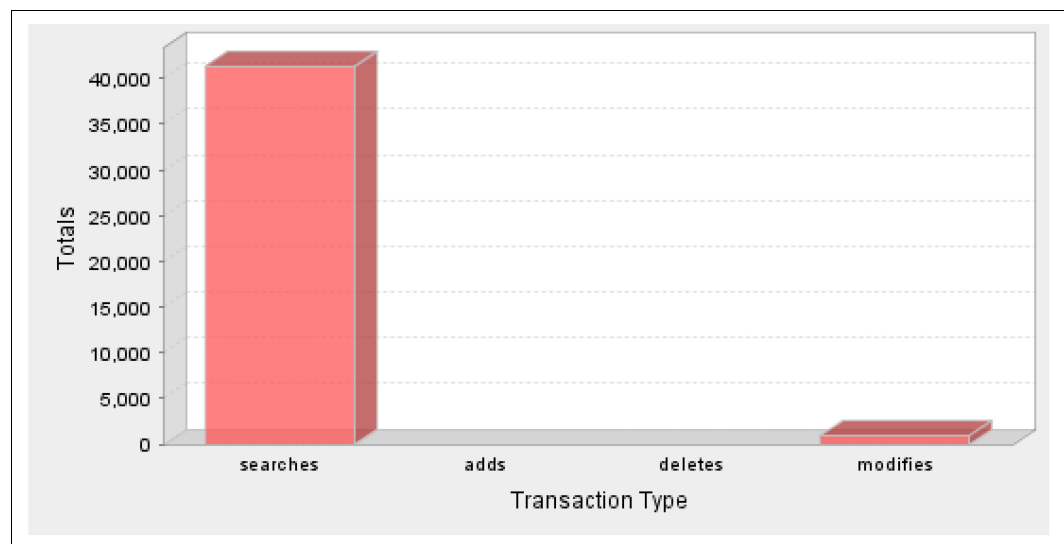


Figure 7-2 Totals by type (without unbinds and binds)

- Distribution by time (all transaction types):

This chart in Figure 7-3 shows, by transaction count, the number of transactions that are completed within a specific time frame. Although the average transaction time for searches is 183 ms, most of transactions completed in 10 ms or less.

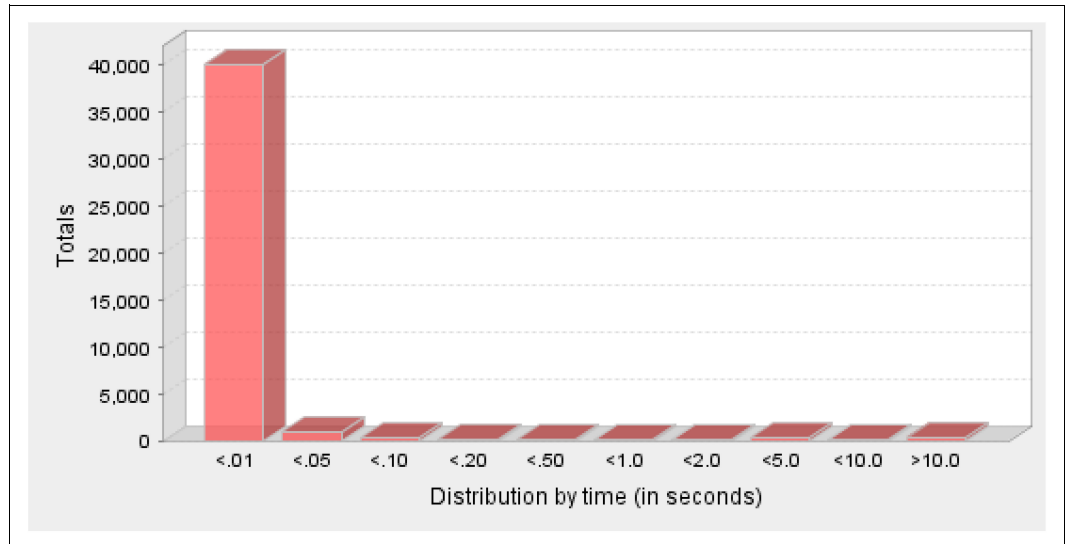


Figure 7-3 Distribution by time

Bind Summary

The Bind Summary section provides a breakdown, by time, of the quantity and distribution 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) indicate problems with directory load, performance, or the network communication (for example, very slow or high error rate link).

► Bind distribution by time:

The chart in Figure 7-4 shows, by transaction count, the number of transactions that completed within a specific time frame.

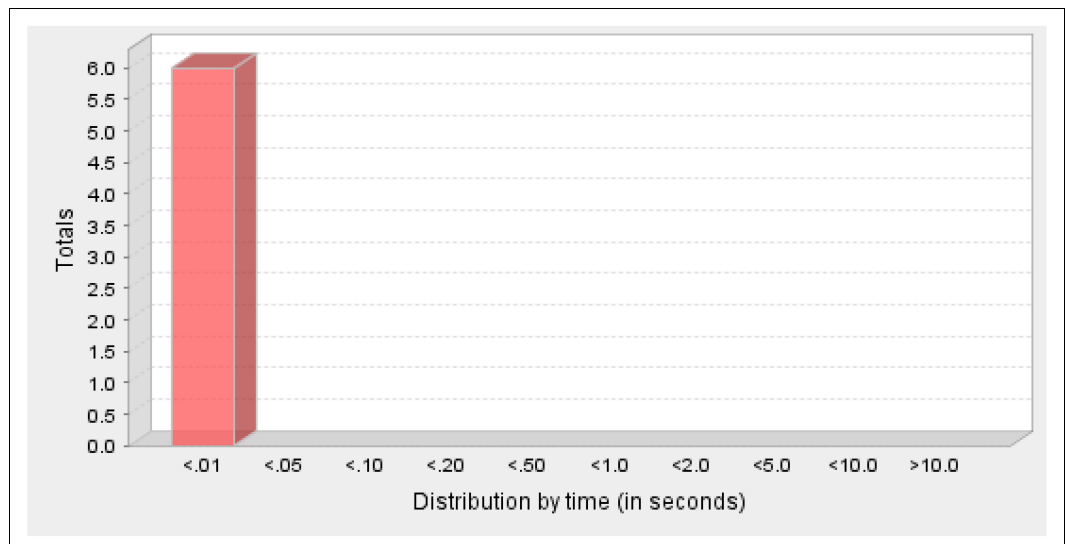


Figure 7-4 Number of transactions that completed within a specific time frame

Unbind Summary

The Unbind Summary section provides a breakdown, by time, of the quantity and distribution 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) indicate problems with the directory load or performance.

Add Summary

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

The chart in Figure 7-5 on page 50 shows, by transaction count, the number of transactions that completed within a specific 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.

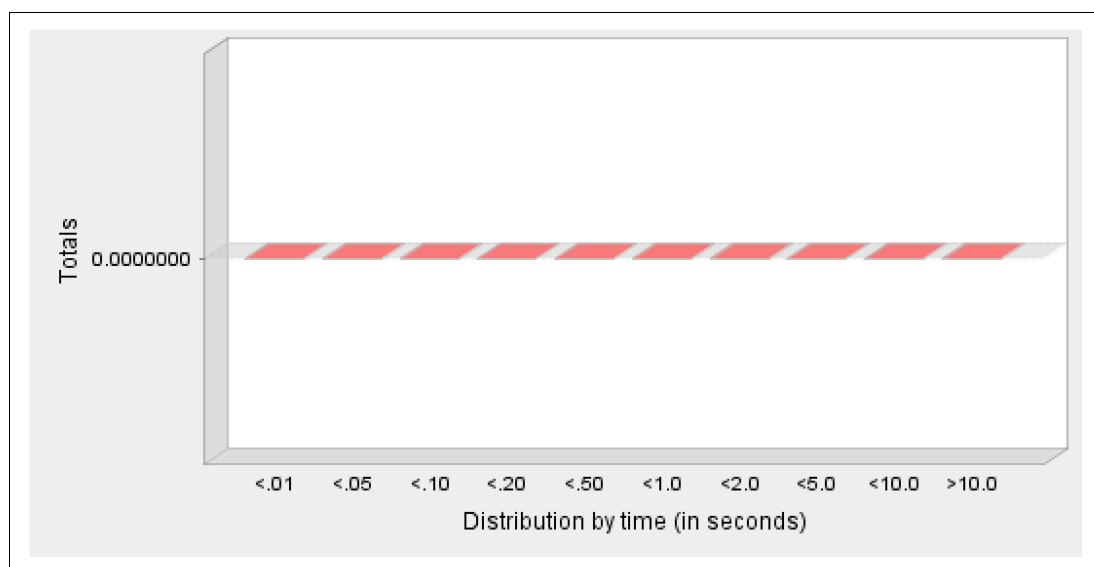


Figure 7-5 Add distribution by time

Search Summary

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

The chart in Figure 7-6 shows by transaction count the number of transactions that completed within a specific time frame.

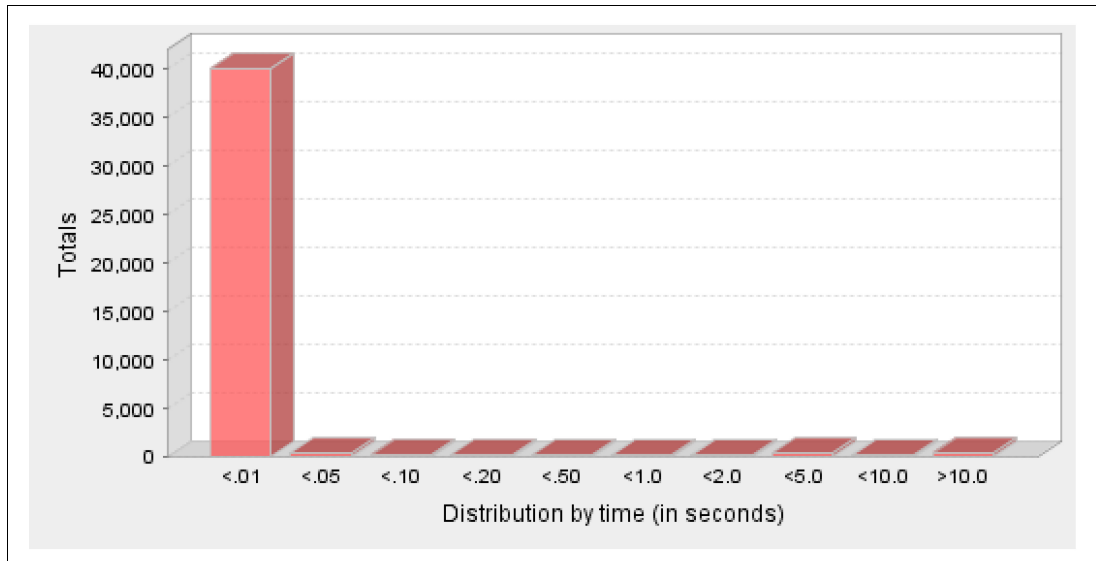


Figure 7-6 Search distribution by time

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

- The longest search text:

The search query that took the longest time is reproduced in the document so that you can evaluate and determine whether tuning or client action must be taken. See Figure 7-7.

```

Longest Search Text
Longest Search (time in Milliseconds) = 15947
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=abocompany,dc=tim
scope: singleLevel
derefAliases: derefAlways
typesOnly: false
filter:
(&(erparent=ERGLOBALID=6776123625579741330,OU=ORGCHART,ERGLOBALID=000000000000
000000000,OU=abocompany,DC=ITIM))((objectclass=ERBPORGITEM)(objectclass=ERLOCATIONI
TEM)(objectclass=ERORGUNITITEM)(objectclass=ERSECURITYDOMAINITEM)))
attributes: erparent, objectclass,

```

Figure 7-7 Longest search text

- Filter attributes used:

This summary of all attributes used in every search transaction found within the audit log can be useful to determine whether indexes must be built or enabled. See Figure 7-8.

Filter Attributes Used	
Total # of Attributes used in search filters =	16
Attribute = epolicytarget	Attribute count = 9772
Attribute = epnerequisite	Attribute count = 21
Attribute = epolicymembership	Attribute count = 11583
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 = 3662
Attribute = ereqpolicytarget	Attribute count = 10077
Attribute = erword	Attribute count = 921
Attribute = erenabled	Attribute count = 10686
Attribute = employmentstatus	Attribute count = 610
Attribute = erdeleted	Attribute count = 24
Attribute = erparent	Attribute count = 14958
Attribute = erservicename	Attribute count = 1831

Figure 7-8 Filter attributes used

Delete Summary

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

The chart in Figure 7-9 shows, by transaction count, the number of transactions that completed within a specific 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.

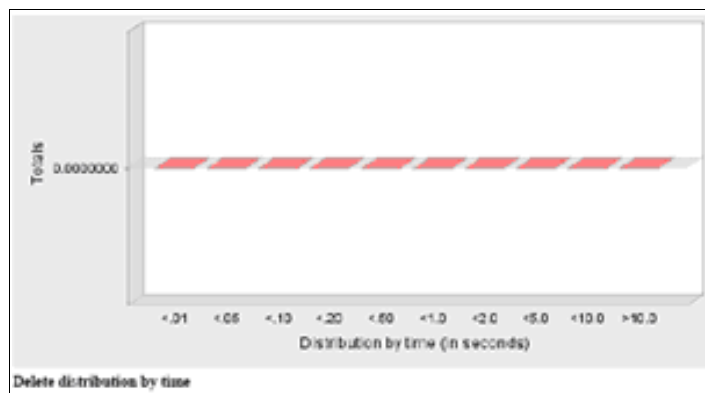


Figure 7-9 Delete distribution by time

Modify Summary

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

The chart in Figure 7-10 on page 53 shows by transaction count the number of transactions that completed within a specific time frame.

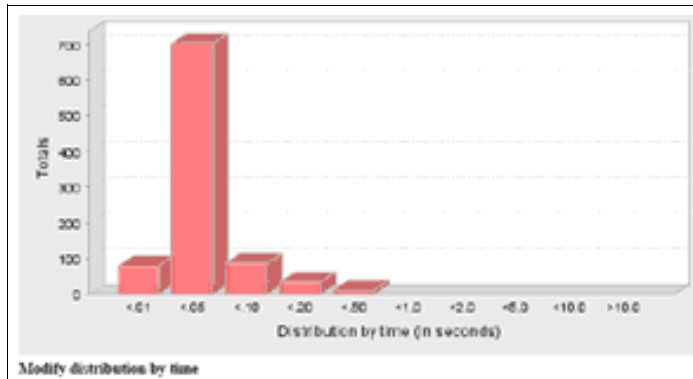


Figure 7-10 Modify distribution by time

7.2 tune_enablemonitor.sh

This script turns on the DB2 monitors in Table 7-3. The monitors need to be running to gather statistics in the database. You can use the following monitor switches to read and analyze this data.

Table 7-3 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 and stop times, statement identification
UOW (unit of work)	DFT_MON_UOW	Start and end times, completion status
TIMESTAMP	DFT_MON_TIMESTAMP	Time stamps

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

With the **tune_enablemonitor.sh** script, you must stop and restart DB2 after you run the script. Make sure that Lightweight Directory Access Protocol (LDAP) is down you run this script. You can bring LDAP back up after you run this script. You own the DB2 instance to run this script.

7.3 perftune_enablemonitor_all.sh

This script turns on the monitor listed in Table 7-3, along with the one in Table 7-4.

Table 7-4 DB2 monitor

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

This monitor affects the performance if it is left running for long periods of time. Turn it on only for troubleshooting and checking the table space.

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

7.4 tune_disablemonitor.sh

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

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

7.5 perfanalyze_indexes.pl

This script determines whether searches of the LDAP server are using filters for attributes that do not have indexes. It does this with 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 directly from the LDAP server
- ▶ Processing the LDAP schema files to read all attributes and determining which have indexes
- ▶ Processing either an audit log or a dynamic SQL snapshot from the database instance to determine all attributes that you are searching
- ▶ 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. If this is not possible or not desirable, you 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 mentions this fact but the index status of these attributes cannot be determined.

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

7.5.1 Use

Starting the script with the `-h` option produces an output with the following use 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 the `ibmslapd.conf` file
 - d** Directory where the 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 the ones not indexed

Note: Use UNIX-style slashes for all files and directories, even on a Microsoft Windows system. If no arguments are given, the program reads input from STDIN.

7.5.2 Examples

On a Windows platform, 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 (the `-a` option shows all attributes, not just unindexed 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 specified directory but also to include a custom file, use this script:

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

To analyze indexes from an audit log that is based on a schema from a remote machine, use this script:

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

7.6 perfanalyze_audit.pl

This script finds metadata from information in an audit log, such as what queries occur most often and how long queries take. It can do this for either the full query with the values searched or in a fuzzy method, where the attributes are retained but the value searched 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 it with the **perfanalyze_indexes.pl** script to locate and index attributes used in long-running LDAP queries.

7.6.1 Use

Starting the script with the `-h` option produces an output with the following use 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, with the following as valid options:
 - 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 the following: second, minute, hour, day, month
 - c** Statements longer than this time are not included in the timings report (default is 0.1)
 - b** Show search bases
 - p** Show search scopes
- Other options:
 - h** Displays this help information
 - i** File containing the log for processing
 - o** File to put the processed results in (default is STDOUT)

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

7.6.2 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
```

7.7 perfanalyze_dynamicsql.pl

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

Use this script to determine which LDAP queries are taking the longest, possibly from missing indexes. Use this 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 full query being printed.

By default, only the queries that take 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 that are being printed.

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

7.7.1 Use

Starting the script with the **-h** option produces output with the following use information:

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

- i** File that contains dynamic SQL statements for processing
- d** Database name to get dynamic SQL statements from directly
- s** If provided, the temporary file with the dynamic SQL is saved
- o** File to put the processed results in (default is STDOUT)
- t** Truncate length statement (default is 90 characters; 0 = do not truncate)
- c** Cut-off time (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 reads from standard input (stdin).

7.7.2 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 that took longer than 0.01 seconds:

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

7.8 perfanalyze_database.pl

This script parses and reports on the database by analyzing database 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 (insofar as that is possible, given the system resources). Using this script enables you to determine

the current hit ratio and the impact that increasing buffer pools might have. The goal is to achieve 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 if you use the **-d** flag. If you want to retain this snapshot, use the **-s** flag also.

7.8.1 Use

Starting the script with the **-h** option produces an output with the following use 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 provided, the temporary file with the snapshot is saved
- o** File to put the processed results in (default is STDOUT)

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

7.8.2 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
```

7.9 perfanalyze_tables.pl

This script parses and reports 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 if you use the **-d** flag. If you want to retain this snapshot, use the **-s** flag also.

7.9.1 Use

Starting 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 provided, the temporary file with the snapshot is 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 reads input from STDIN.

7.9.2 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 directly from the database):

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




Why you must use runstats

IBM DB2 software uses a sophisticated set of algorithms to optimize the access to data that is 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 that is maintained by the database manager.

In addition, Directory Server creates several indexes for tables in the database. These indexes are used to minimize the data that is accessed to locate a particular row in a table.

In a read-only environment, the distribution of the data changes very little. However, with updates and additions to the database, it is not uncommon for the distribution of the data to change significantly. Similarly, it is possible for data in tables to become ordered in an inefficient way.

To remedy these situations, Directory Server provides tools to help optimize the access to data by updating the statistics and to reorganize the data within the tables of the database. The **idsrunstats** utility that it includes takes into account specific environment variables (**LDAP_MAXCARD** or **IBMSLDAP_USE_SELECTIVITY** if set) and runs a DB2 **runstats** command. Starting with Directory Server Version 6.2, this utility can be run offline or online. If you choose to run **idsrunstats** online, do that during a period of minimal activity to reduce the potential for impact on the environment. It is also important to consider that some SQL statements might be cached by the Directory Server and cannot benefit from the new statistics until you restart **ibmslapd**.

In general, use the **idsrunstats** utility either at a regular interval or after the following events:

- ▶ After heavy database updates, such as additions, modifications, deletes, or LDAP Directory Interchange Format (LDIF) imports
- ▶ After reorganizing a table
- ▶ After a new index is defined

In addition to periodically using **idsrunstats**, there are times when a DB2 reorganization is also required to achieve optimal performance. The DB2 **reorg** command is covered in Chapter 9, “When and how to run reorg” on page 67.

8.1 Optimization

The optimizer uses the catalog tables from a database to get information about the database, the amount of data in it, and other characteristics. Then it 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 that is based on inaccurate default statistics.

Optimizing the database updates the statistics that are related to the data tables, which improves performance and query speed. Optimizing the database periodically or after heavy database updates is important for optimal performance. You can do that by using the **idsrunstats** and **reorg** commands to tune the organization of the DB2 data.

The **idsrunstats** command updates statistical information for the DB2 optimizer to improve performance and reports statistics on the organization of the database tables.

The **idsrunstats** command runs DB2 runstats but also artificially inflates the cardinality of the LDAP_DESC table to influence the DB2 optimization. As a result, DB2 attempts to resolve all other attribute filters before looking at the LDAP_DESC table to resolve subtree relationships. Starting with Directory Server Version 6.2, the IBMSLDAPD_USE_SELECTIVITY variable was introduced so that DB2 Selectivity can be used in a Directory Server environment. If DB2 Selectivity is enabled and the IBMSLDAPD_USE_SELECTIVITY is set, the **idsrunstats** command runs DB2 runstats but does not artificially inflate the cardinality of the LDAP_DESC table. DB2 Selectivity requires the actual cardinalities of the tables to be used to benefit from the parameter. LDAP_MAXCARD and DB2 Selectivity are explained further in Chapter 6, “Using LDAP_MAXCARD and IBMSLDAPD_USE_SELECTIVITY” on page 37.

If the Directory Server environment is used by another application, it is important to review the application’s recommendations for tuning and maintenance. Typically, it is best to follow the guidelines for the application that uses the Directory Server environment so that it is tuned appropriately for the applications’ workload.

8.2 Which runstats to use

There are many methods available for using **runstats**. DB2 provides the **runstats** and **reorgchk** utilities, but many other applications provide their own versions. As previously mentioned, if the Directory Server environment is being used by a specific application, it is best to follow that application’s guidelines so that the Directory Server environment is tuned appropriately for the expected workload.

The **idsrunstats** utility provided by the Directory Server runs the DB2 **runstats** and artificially inflates the cardinality of the LDAP_DESC table to 9E18. If you are using Directory Server Version 6.2 or later and DB2 Selectivity is enabled, the **idsrunstats** utility detects this and the cardinality is not artificially inflated. This allows the DB2 optimizer to use the actual statistics to narrow the search results before reading the table that identifies the entries that are descendants of a base in a search. For more information about DB2 Selectivity, see Chapter 6, “Using LDAP_MAXCARD and IBMSLDAPD_USE_SELECTIVITY” on page 37.

There are two Directory Server methods that can be used to optimize the database. The GUI method is provided through the Directory Server Configuration Tool (**idsxcfg**). The command-line method is through the **idsrunstats** command. The **idsxcfg** GUI utility uses the **idsrunstats** command to update statistical information that is used by the query optimizer for all of the LDAP tables.

In DB2 Version 9 and later, there is also an option to use DB2 automatic **runstats** with Directory Server. For more information about that, see 8.3, “DB2 automatic statistics collection” on page 63.

Use the Configuration Tool to optimize the database

Follow these steps to optimize the database by using the Configuration Tool (`idsxcfg`):

1. Start the Configuration Tool by typing `idsxcfg` on the command line.
2. Click **Optimize database** at the left side of the window.
3. In the “Optimize database” window, click **Optimize**.
4. After a message indicates that the database was successfully optimized, you must restart the server for the changes to take effect.

Use a command to optimize the database

To optimize the database by using the command line, run the following command:

```
idsrunstats -I <instance_name>
```

Where `<instance_name>` is an optional parameter.

Running `idsrunstats` while the server is running: Starting with Directory Server Version 6.2 and later, it is possible to run **idsrunstats** while the server is running. However, this can cause inconsistent statistics to occur because updates are occurring on a table while statistics are being generated. This can negatively affect the performance of the environment. There might also be some SQL statements that are cached by the Directory Server that cannot benefit from the new statistics until you restart **ibmslapd**.

8.3 DB2 automatic statistics collection

Automatic statistics collection, also known as *auto-runstats*, was introduced in DB2 Version 8.2 as part of DB2 automated table maintenance feature. This section focuses on DB2 Version 9 and later. One concern about **runstats** is how often to run it in a particular environment. The answer to that question varies, based on the activities that are performed in the environment. One way to reduce the administration for keeping the table statistics up to date is to use the auto-runstats feature of DB2 rather than the **idsrunstats** command, as described in 8.2, “Which runstats to use” on page 62. With automatic statistics collection, you can let DB2 determine whether database statistics need to be updated. DB2 automatically runs the **runstats** utility in the background to ensure that the most current database statistics are available.

There are some considerations to be aware of, though. If you are using DB2 Selectivity in your Directory Server environment, enabling auto-runstats is straight-forward. However, if you are using **LDAP_MAXCARD**, you must artificially inflate the cardinality on the **LDAP_DESC** to 9E18 and exclude the **LDAP_DESC** from auto-runstats. Again, if there are specific application recommendations, follow those to ensure optimal performance.

Parameter descriptions

Take a closer look at some of the parameter details in the subsections that follow.

auto_maint

Automatic maintenance configuration parameter. This parameter is the parent of all of the other automatic maintenance database configuration parameters:

- ▶ **auto_db_backup**
- ▶ **auto_tbl_maint**
- ▶ **auto_runstats**

auto_stats_prof

- ▶ **auto_stmt_stats**
- ▶ **auto_prof_upd**
- ▶ **auto_reorg**

auto_tbl_maint

This parameter is the parent of all table maintenance parameters (**auto_runstats**, **auto_stats_prof**, **auto_prof_upd**, and **auto_reorg**). When this parameter is disabled, all of its child parameters are also disabled. However, their settings, as recorded in the database configuration file, do not change. When this parent parameter is enabled, recorded values for its child parameters take effect. In this way, table maintenance can be enabled or disabled globally.

By default, this parameter is set to ON.

auto_runstats

This automated table maintenance parameter enables or disables automatic table runstats operations for a database. A runstats policy (a defined set of rules or guidelines) can be used to specify the automated behavior. Statistics that are collected by the **runstats** utility are used by the optimizer to determine the most efficient plan for accessing the physical data. To be enabled, this parameter must be set to ON, and its parent parameters must also be enabled.

By default, this parameter is set to ON.

auto_stats_prof

When enabled, this automated table maintenance parameter turns on statistical profile generation to improve applications with workloads that include complex queries, many predicates, joins, and grouping operations over several tables. To be enabled, this parameter must be set to ON, and its parent parameters must also be enabled.

By default, this parameter is set to OFF.

auto_stmt_stats

This parameter enables and disables the collection of real-time statistics. It is a child of the **auto_runstats** configuration parameter. This feature is enabled only if the parent, **auto_runstats** configuration parameter, is also enabled. For example, to enable **auto_stmt_stats**, set **auto_maint** and **auto_tbl_maint**, and set **auto_runstats** to ON. To preserve the child value, the **auto_runstats** configuration parameter can be ON while the **auto_maint** configuration parameter is OFF. The corresponding auto-runstats feature is still OFF.

To enable automatic statistics collection, from the command line, set each of the following configuration parameters to ON (see the example that follows):

- ▶ **AUTO_MAINT**
- ▶ **AUTO_TBL_MAINT**
- ▶ **AUTO_RUNSTATS**

Example 8-1 Commands to enable automatic collection of statistics

```
su - <instance owner>
db2start
db2 connect to <database>
db2 update db config using AUTO_MAINT ON AUTO_TBL_MAINT ON AUTO_RUNSTATS ON
db2 connect reset
```

Optional: To enable the automatic statistics profile generation, set the following two configuration parameters to ON:

- ▶ **AUTO_STATS_PROF**
- ▶ **AUTO_PROF_UPD**

Optional: To enable real-time statistics gathering, set the **AUTO_STMT_STATS** configuration parameter to ON. If this configuration parameter is set to ON, table statistics are automatically compiled at statement compilation time whenever they are needed to optimize a query.



When and how to run reorg

As you learned in Chapter 8, “Why you must use runstats” on page 61, **runstats** is used to update the statistical information that is used by the query optimizer for all the LDAP tables. In an environment where the data distribution changes, reorgs or table reorganization might be required to achieve optimal performance. Table reorganization is used to defragment a table, reclaim free space, and eliminate overflow rows to improve data access performance. Optionally, it can be used to reorder data in a particular index, which helps optimize queries that use that index. The organization of the data is addressed with the **reorgchk** and **reorg** commands.

The **reorgchk** command updates statistical information to the DB2 optimizer to improve performance, and reports statistics on the organization of the database tables. If you plan on performing a **reorgchk**, running **runstats** is unnecessary.

By using the data that is generated by **reorgchk**, the **reorg** command reorganizes table spaces to improve access performance and reorganizes indexes so that they are more efficiently clustered. The **reorgchk** and **reorg** commands can improve both search and update operation performance. Next, we’ll delve into these details:

- ▶ Performing a reorg when necessary
- ▶ DB2 automatic table and index reorganization

9.1 Performing a reorg when necessary

After you generate organizational information about the database by using **reorgchk** for reorganization, **reorg** finds the necessary tables and indexes and attempts to reorganize them. This can take a long time. The time that it takes for the reorganization process increases as the DB2 database size increases. This step is necessary if **reorgchk** or **runstats** does not achieve the required results.

9.1.1 Performing a reorgchk

After DB2 database updates, table indexes can become inefficient or unreliable and performance can degrade. Correct this situation by performing a DB2 **reorgchk**, where *ldapdb2* is a placeholder for the name of your database:

```
db2 connect to ldapdb2
db2 reorgchk update statistics on table all
```

To generate a **reorgchk** output file, which is advisable if you plan to run the **reorg** command, add the name of the file to the end of the command, for example:

```
db2 reorgchk update statistics on table all > reorgchk.out
```

Example 9-1 shows a sample **reorgchk** report.

Example 9-1 Sample reorgchk report

```
db2 => reorgchk current statistics on table all
```

Table statistics:

```
F1: 100 * OVERFLOW / CARD < 5
F2: 100 * TSIZE / ((FPAGES-1) * (TABLEPAGESIZE-76)) > 70
F3: 100 * NPAGES / FPAGES > 80
```

CREATOR	NAME	CARD	OV	NP	FP	TSIZE	F1	F2	F3	REORG
LDAPDB2	ACLPERM	2	0	1	1	138	0	-	100	---
LDAPDB2	ACLPROP	2	0	1	1	40	0	-	100	---
LDAPDB2	ALIASEDOBJECT	-	-	-	-	-	-	-	-	---
LDAPDB2	AUDIT	1	0	1	1	18	0	-	100	---
LDAPDB2	AUDITADD	1	0	1	1	18	0	-	100	---
LDAPDB2	AUDITBIND	1	0	1	1	18	0	-	100	---
LDAPDB2	AUDITDELETE	1	0	1	1	18	0	-	100	---
LDAPDB2	AUDITEXTOPEVENT	1	0	1	1	18	0	-	100	---
LDAPDB2	AUDITFAILEDOPONLY	1	0	1	1	18	0	-	100	---
LDAPDB2	AUDITLOG	1	0	1	1	77	0	-	100	---
...										
SYSIBM	SYSINDEXCOLUSE	480	0	6	6	22560	0	100	100	---
SYSIBM	SYSINDEXES	216	114	14	28	162216	52	100	50	*-*
...										
SYSIBM	SYSPLAN	79	0	6	6	41554	0	100	100	---

SYSIBM	SYSPLANAUTH	157	0	3	3	9106	0	100	100	---
SYSIBM	SYSPLANDEP	35	0	1	2	5985	0	100	50	--*

Index statistics:

F4: CLUSTERRATIO or normalized CLUSTERFACTOR > 80

F5: $100 * (KEYS * (ISIZE+8) + (CARD-KEYS) * 4) / (NLEAF * INDEXPAGESIZE) > 50$

F6: $(100-PCTFREE) * (INDEXPAGESIZE-96) / (ISIZE+12) ** (NLEVELS-2) * (INDEXPAGESIZE-96) / (KEYS * (ISIZE+8) + (CARD-KEYS) * 4) < 100$

CREATOR	NAME	CARD	LEAF	LVLS	ISIZE	KEYS	F4	F5	F6	REORG
---------	------	------	------	------	-------	------	----	----	----	-------

Table: LDAPDB2.ACLPERM

LDAPDB2	ACLPERM_INDEX	2	1	1	6	2	100	-	-	---
---------	---------------	---	---	---	---	---	-----	---	---	-----

Table: LDAPDB2.ACLPROP

LDAPDB2	ACLPROP_INDEX	2	1	1	6	2	100	-	-	---
---------	---------------	---	---	---	---	---	-----	---	---	-----

Table: LDAPDB2.ALIASEDOBJECT

LDAPDB2	ALIASEDOBJECT	-	-	-	-	-	-	-	-	---
---------	---------------	---	---	---	---	---	---	---	---	-----

LDAPDB2 ALIASEDOBJECTI

LDAPDB2	ALIASEDOBJECTI	-	-	-	-	-	-	-	-	---
---------	----------------	---	---	---	---	---	---	---	---	-----

LDAPDB2 RALIASEDOBJECT

LDAPDB2	ALIASEDOBJECTI	-	-	-	-	-	-	-	-	---
---------	----------------	---	---	---	---	---	---	---	---	-----

Table: LDAPDB2.AUDIT

LDAPDB2	AUDITI	1	1	1	4	1	100	-	-	---
---------	--------	---	---	---	---	---	-----	---	---	-----

Table: LDAPDB2.AUDITADD

LDAPDB2	AUDITADDI	1	1	1	4	1	100	-	-	---
---------	-----------	---	---	---	---	---	-----	---	---	-----

Table: LDAPDB2.AUDITBIND

LDAPDB2	AUDITBINDI	1	1	1	4	1	100	-	-	---
---------	------------	---	---	---	---	---	-----	---	---	-----

Table: LDAPDB2.AUDITDELETE

LDAPDB2	AUDITDELETEI	1	1	1	4	1	100	-	-	---
---------	--------------	---	---	---	---	---	-----	---	---	-----

Table: LDAPDB2.AUDITEXTOPEVENT

...

Table: LDAPDB2.SN

LDAPDB2	RSN	25012	148	2	14	25012	99	90	0	---
---------	-----	-------	-----	---	----	-------	----	----	---	-----

LDAPDB2	SN	25012	200	3	12	25012	99	61	119	--*
---------	----	-------	-----	---	----	-------	----	----	-----	-----

LDAPDB2	SNI	25012	84	2	4	25012	99	87	1	---
---------	-----	-------	----	---	---	-------	----	----	---	-----

...

Table: LDAPDB2.TITLE

LDAPDB2	TITLEI	-	-	-	-	-	-	-	-	---
---------	--------	---	---	---	---	---	---	---	---	-----

Table: LDAPDB2.UID

LDAPDB2	RUID	25013	243	3	17	25013	0	62	79	*--
---------	------	-------	-----	---	----	-------	---	----	----	-----

LDAPDB2	UID	25013	273	3	17	25013	100	55	79	---
---------	-----	-------	-----	---	----	-------	-----	----	----	-----

LDAPDB2	UIDI	25013	84	2	4	25012	100	87	1	---
---------	------	-------	----	---	---	-------	-----	----	---	-----

Table: LDAPDB2.UNIQUEMEMBER

LDAPDB2	RUNIQUEMEMBER	10015	224	3	47	10015	1	60	44	*--
---------	---------------	-------	-----	---	----	-------	---	----	----	-----

LDAPDB2	UNIQUEMEMBER	10015	284	3	47	10015	100	47	44	*--
---------	--------------	-------	-----	---	----	-------	-----	----	----	-----

LDAPDB2	UNIQUEMEMBERI	10015	14	2	4	7	100	69	8	---
---------	---------------	-------	----	---	---	---	-----	----	---	-----

...

Table: SYSIBM.SYSFUNCTIONS

SYSIBM	IBM127	141	1	1	13	141	65	-	-	*--
--------	--------	-----	---	---	----	-----	----	---	---	-----

SYSIBM	IBM25	141	2	2	34	141	100	72	60	---
--------	-------	-----	---	---	----	-----	-----	----	----	-----

SYSIBM	IBM26	141	2	2	32	141	78	68	63	*--
--------	-------	-----	---	---	----	-----	----	----	----	-----

SYSIBM	IBM27	141	1	1	23	68	80	-	-	*--
--------	-------	-----	---	---	----	----	----	---	---	-----

SYSIBM	IBM28	141	1	1	12	2	99	-	-	---
SYSIBM	IBM29	141	1	1	4	141	100	-	-	---
SYSIBM	IBM30	141	3	2	59	141	78	76	38	*--
SYSIBM	IBM55	141	2	2	34	141	99	72	60	---
...										

CLUSTERRATIO or normalized CLUSTERFACTOR (F4) will indicate 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 may be flagged as needing REORG. Specify the most important index for REORG sequencing.

Using the statistics generated by **reorgchk**, run **reorg** to update database table organization, as shown in 9.1.2, “Reorganize a table” on page 70.

Keep in mind that you need to run **reorgchk** periodically. For example, **reorgchk** might need to be run after a large number of updates have been performed. LDAP tools such as **ldapadd**, **ldif2db**, and **bulkload** can potentially do large numbers of updates that require a **reorgchk**. The performance of the database should be monitored and a **reorgchk** performed when performance starts to degrade.

A **reorgchk** must be performed on all LDAP replicas because each replica uses a separate database. The LDAP replication process does not include the propagation of database optimizations.

Because LDAP caches prepared DB2 statements, you must stop and restart **ibmslapd** for DB2 changes to take effect.

9.1.2 Reorganize a table

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

1. Stop the **ibmslapd** process:

```
ibmslapd -I <instance name> -k
```

2. Issue the following commands to reorg a table:

```
su - <instance owner>
db2start
db2 connect to <database name>
db2 reorg table <table_name>
```

Example:

```
su - ldapdb2
db2start
db2 connect to ldapdb2
db2 reorg table LDAPDB2.SN
```

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

```
su - ldapdb2
$ ./reorgchk
$ exit
```

4. Restart the **ibmslapd** process:

```
$ ibmslapd -I <instance name>
```

9.1.3 Reorganize an index

Follow the steps in this section to reorganize database indexes that have an asterisk in the last column.

1. Stop the **ibmslapd** process:

```
ibmslapd -I <instance name> -k
```

2. Perform a **reorg** either on a table to match the order of a particular index or on the index:

- a. Run the following commands to reorganize a table to match the order of a particular index:

```
su - <instance owner>
db2start
db2 connect to <database name>
db2 reorg table <table_name> index <index_name>
```

<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.SN

LDAPDB2.SNI

After:

LDAPDB2.SN = <table_name>

LDAPDB2.SNI = <index_name>

Example:

```
su - ldapdb2
db2start
db2 connect to ldapdb2
db2 reorg table LDAPDB2.SN index LDAPDB2.SNI
```

- b. Run the following commands to reorganize indexes with an asterisk in the last column:

```
su - <instance owner>
db2start
db2 connect to <database name>
db2 reorg index <index_name>
```

Example:

```
su - ldapdb2
db2start
db2 connect to ldapdb2
db2 reorg index LDAPDB2.RTELEPHONENUMBER
```

3. After all reorgs are finished, run the following script (required):

```
su - ldapdb2
$ ./reorgchk
$ exit
```

4. Restart the **ibmslapd** process:

```
$ ibmslapd -I <instance name>
```

The following three indexes (or any other index not listed here that is created in these three tables) should never need to be reorganized, because it requires a 32 k temp page area to do and does not result in any benefit. 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
```

Generally speaking, because most data in LDAP is accessed by index, reorganizing tables is usually not as beneficial as reorganizing indexes.

9.1.4 Guidelines for performing a reorganization

If the number in the column that has an asterisk is close to the recommended value that is described in the header of each section and one reorganization has already been attempted, you can probably skip a reorganization on that table or index.

In the table LDAPDB2.LDAP_ENTRY there is a LDAP_ENTRY_TRUNC index and a SYSIBM.SQL index. Give preference to the SYSIBM.SQL index if attempts to reorganize them seem to alternate between one or the other needing reorganization.

When an attribute length is defined to be less than or equal to 240 bytes, the attribute table contains three columns:

- ▶ EID
- ▶ Attribute
- ▶ Reversed attribute

In this case, the forward index is created by using the EID and attribute columns as index keys. For example, the **SN** attribute is defined to have the maximum length, which is less than or equal to 240 bytes. So the attribute table contains the EID, SN, and RSN columns, and the following indexes are created for this attribute table:

LDAPDB2.RSN	A reverse index where the defined index keys are the EID and RSN columns
LDAPDB2.SN	A forward index where the defined index keys are the EID and SN columns
LDAPDB2.SNI	An update index where the defined index key is the EID column

Reorganize all of the attribute tables that you want to use in searches. In most cases you will want to reorganize to the forward index, but in cases with searches that begin with an asterisk (*), reorganize to the reverse index.

When an attribute length is defined to be greater than 240 bytes, the attribute table contains four columns:

- ▶ EID
- ▶ Attribute
- ▶ Truncated attribute
- ▶ Reversed truncated attribute

In this case, the forward index is created using the EID and truncated attribute columns as index keys. For example, the **CN** attribute is defined to have the maximum length, which is

greater than 240 bytes, so the attribute table contains the EID, CN, CN_T, and RCN_T columns and the following indexes are created for this attribute table:

LDAPDB2.RCN	A reverse index where the defined index keys are the EID and RCN_T columns
LDAPDB2.CN	A forward index where the defined index keys are the EID and CN_T columns
LDAPDB2.CNI	An update index where the defined index key is the EID column

9.2 DB2 automatic table and index reorganization

Automatic table maintenance or *auto_reorg* is part of the hierarchy of automatic maintenance database configuration parameters. For more information about these parameters, see 8.3, “DB2 automatic statistics collection” on page 63. Automatic reorganization is based on reorganizing the tables by using **reorgchk**. If you plan to use automatic reorganization, it is best to also enable automatic collection of statistics so that the recommendations made after the **reorgchk** are based on the most up-to-date statistics.

Automatic reorganization behavior is controlled through a maintenance policy that specifies the following items:

- ▶ An offline maintenance window. This is a time span during which automatic reorganization will perform offline table and index reorganizations. This is the only part of the maintenance policy that does not have a default value. Tables are taken offline during this time (no write access allowed to the tables and no read access during the REPLACE phase of reorganization), so the requirements for this window are different for each user. An offline window must be specified to use the automatic reorganization feature. If no offline window is specified and automatic reorganization is enabled, DB2 fails each attempt to automatically reorganize a table.
- ▶ A set of tables to be considered for reorganization. These tables can be limited by size or name. By default, all user tables are considered for automatic reorganization, regardless of table size.
- ▶ Options that modify the reorganization operation when it is applied to a table. For example, this could be to keep or rebuild compression dictionaries when performing table reorganization.
- ▶ An online maintenance window. In DB2 9, automatic reorganization of indexes can be performed online (read and write access to the indexes is allowed during reorganization). If automatic online index reorganization is configured, the online maintenance window is the window of time during which the online index reorganizations are to be performed.

Note:

The maintenance policy can be updated only by using the DB2 Control Center graphical interface. There is no command line or programmatic interface.

9.2.1 Parameter description

Take a closer look at the **auto_reorg** parameter.

This maintenance parameter enables or disables automatic table and index reorganization for a database. A reorganization policy (a defined set of rules or guidelines) can be used to specify the automated behavior. To be enabled, this parameter must be set to ON, and its parent parameters must also be enabled. By default, this parameter is set to OFF.

To enable automatic reorganization, use the command line to set each of the following configuration parameters to ON (also see Example 9-2):

- ▶ **AUTO_MAINT**
- ▶ **AUTO_TBL_MAINT**
- ▶ **AUTO_RUNSTATS**
- ▶ **Auto_REORG**

Example 9-2

```
su - <instance owner>
db2start
db2 connect to <database>
db2 update db config using AUTO_MAINT ON AUTO_TBL_MAINT ON AUTO_RUNSTATS ON
AUTO_REORG ON
db2 connect reset
```

In addition to enabling the automatic reorganization in the database configuration, be sure to configure an offline maintenance window by using the DB2 Control Center.



LDAP searches and slow operations

In this chapter, we describe how to improve LDAP searches and how to identify operations that are slow. We cover the following situations:

- ▶ Improving LDAP searches
- ▶ Identifying slow operations

10.1 Improving LDAP searches

Some LDAP searches are inherently slow and expensive. Consider this example:

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

IBM DB2 software 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, such as this example, are slow:

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

Searches with NOT clauses are slow. Recent fix packs for IBM 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 previous one or this example:

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

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

Another search shows a related issue:

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

This search references the object class table three times. Avoid references to the object class table 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 (if `partType` is indexed) because the `partType` table, like the table for every attribute type, is much smaller than the object class table. The `(objectclass=*)` search clause is at no cost. The directory recognizes that every entry satisfies this filter; therefore, it generates no SQL for this clause. The `(objectclass=top)` search clause is expensive, although it returns the same result set. The Directory Server generates SQL for this clause, and it 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 particular 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.

It is best to index every attribute type that is used in a search filter. If the table is not indexed, the search forces a table scan, which is always bad for 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 on the application side. Consider this search:

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

If only a few entries have values for the `deleted` and `suspended` attributes, it is much faster to filter them on the application side.

IBM Directory Server supports alias dereferencing so that entries in the directory can be known by several names. Alias dereferencing is always slow. Do not use aliases in large directories, and turn off alias dereferencing in the directory.

10.2 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 by, for example, indexing an attribute type that is used in a search filter.

To identify slow operations, run the audit log and log all operations. The audit log records data for each operation, with time stamps for both the request and response. By comparing the time stamps, you can identify the particular operations that are slow.

Example 10-1 shows a sample of an audit log data with response times.

Example 10-1 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)
```

By subtracting the operation completion time of 2005-09-15-05:50:33.398 from the operation start time of 2005-09-15-05:50:33.399, we get an execution time of 1 millisecond, so this search is almost instantaneous. However, the search that is shown in Example 10-2, takes almost 7 seconds (2005-09-15-07:21:55.052 - 2005-09-15-07:21:48.135).

Example 10-2 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 that takes more than 100 milliseconds is a problem search and needs to be corrected. In many cases, appropriate tuning, such as a change to the indexes, corrects the problem. In a few cases, a change within the application might be required.

10.2.1 Auditing for performance

To identify performance bottlenecks during operations, you can check the server audit log for the summary figures that indicate performance hotspots. The following hotspots are identified for auditing:

- ▶ When an operation must wait in the worker thread queue for a long time before the worker thread starts running the operation.
- ▶ The time that is spent for cache contention inside the back end needs to be tracked.

- The time that is spent in handling client I/O, that is, the time spent in receiving the request and returning the result. This value can also be used for detecting bottlenecks because of slow clients or network issues.

Using the audited performance hotspot data, directory administrators can use the system audit facilities to log the LDAP audit record with the system-defined record format.

While you are auditing the performance profiling, consider the following points:

- The configuration options can be enabled to auditing for a combination of different types of operations, for example, auditing for Add and Modify operations only, along with the auditing for performance.
- At the end of operation execution, the audit information is stored in the server audit logs only. In a scenario where the server is having performance bottlenecks and is in a hung state, the `cn=workers, cn=monitor` search can be issued. This search gives information about where each worker is stuck. It is obtained by accumulating information that is collected about the worker until that point in the audit records.

For each operation, performance data field in the audit records is controlled by using the `ibm-auditPerformance` configuration option. The following performance data fields are defined for each operation:

- **operationResponseTime**

This field represents the time difference in milliseconds between the time the operation was received and the time its response was sent. The operation received time and the response sent time of an operation are published in audit v3 header.

- **timeOnWorkQ**

This field represents time in milliseconds spent in the worker queue before execution is initiated on the operation. The value of this field is the difference between the time execution was initiated and the time the operation was received.

- **rdbmLockWaitTime**

This field represents time in milliseconds spent in acquiring locks over relational database management system (RDBM) caches during operation execution. The value in this field helps administrators determine the time that is spent for cache contention as compared to real work.

The lock wait time over the following resources is also considered:

- Resource cache
- DN cache
- Entry cache
- Filter cache
- Deadlock detector
- RDBM locks
- Attribute cache

Note:

Starting with the Directory Server 6.3 release, attribute cache is deprecated, so avoid using it hereafter.

This is implemented by introducing a field in the operation structure, which is updated when acquiring a lock is attempted during operation execution. In addition, wrapper functions are introduced for functions that attempt to acquire locks over RDBM caches. These wrapper functions take another operation pointer as parameter and update the operation's lock wait time field if `ibm-auditPerformance` is enabled.

► **clientIOTime**

This field represents the time in milliseconds that was spent in receiving the complete operation request and returning the complete operation response. This field is implemented in the operation structure and is updated upon receiving the complete Basic Encoding Rules (BER) for operation request and on successfully returning the response BER message for the operation.

Sample output of the audit Version 3 format for search operation with **ibm-auditPerformance** enabled is shown in Example 10-3.

Example 10-3 Sample output with ibm-auditPerformance enabled

```
AuditV3--2006-09-09-10:49:01.863-06:00DST--V3 Search--
bindDN: cn=root--client: 127.0.0.1:40722--connectionID: 2--
received: 2006-09-09-10:49:01.803-06:00DST--Success
controlType: 1.3.6.1.4.1.42.2.27.8.5.1
criticality: false
base: o=sample
scope: wholeSubtree
derefAliases: neverDerefAliases
typesOnly: false
filter: (&(cn=C*)(sn=A*))
operationResponseTime: 591
timeOnWorkQ: 1
rdbmLockWaitTime: 0
clientIOTime: 180
```

To control server performance hits while collecting information for performance data fields, the **ibm-auditPerformance** field is introduced in the audit configuration section. The value of the field is `False` by default. Therefore, no performance data is collected and published by default. When the value of the **ibm-auditPerformance** field is set to `True`, performance data is collected and published in the audit logs for each operation that is enabled for audit. If the **ibm-auditPerformance** field is enabled, that is, set to `True` in the audit record section, the four performance data fields are audited:

- `operationResponseTime`
- `timeOnWorkQ`
- `rdbmLockWaitTime`
- `clientIOTime`

The values of these performance data fields are times in milliseconds.

10.2.2 Tracing performance

The Directory Server provides information about the runtime performance of the server by using a performance trace that is based on the independent trace facility (**ldtrc**).

The performance profile information in the trace helps you diagnose performance problems. By using the independent trace facility, performance profiling is accomplished with minimum impact on server performance. The independent trace facility profiles operation performance that consists of time stamps at key points that are traversed during an operation execution for a running server instance.

The time stamps such as those in this list are profiled during different stages:

- ▶ RDBM search processing
- ▶ RDBM bind processing
- ▶ RDBM compare processing
- ▶ RDBM write processing

Time stamp details: Time stamp collection points for individual operations are provided only for the RDBM back end.

The **ibm-slapdStartupTraceEnabled** instance configuration option governs the tracing of performance records at server startup. With dynamic tracing (**ldaptrace** client utility), the independent trace utility can be made to start or stop collecting performance records after server startup.

Complete these tasks to activate tracing of performance records dynamically:

1. Activate tracing for performance records by issuing the **ldaptrace** command in the following format:

```
ldaptrace -h <hostname> -p <port number> -D <adminDN> -w <adminpwd> -l on \ -t  
start -- -perf
```

2. Dump the trace to a binary trace file by issuing the following command:

```
ldtrc dmp trace.bin
```

3. Format the trace by issuing the following command:

```
ldtrc fmt trace.bin trace.txt
```

4. After formatting the trace you can analyze the trace and diagnose performance problems. To turn off tracing, issue the following command:

```
ldtrc off
```

Example 10-4 shows a formatted performance trace.

Example 10-4 Formatted performance trace

```
prf_entry LD PERF FrontEnd::operation_in_workQ (3.100.98.1.1.0)  
pid 10255; tid 1167334320; sec 1159183071; nsec 84815000  
En-queue bind op; Worker thread ID 1133718448;  
Work Q size now = 1; client conn (9.124.231.39: conn ID 1)
```



Indexes and direct I/O operations

In this chapter, we provide an overview of indexes and direct I/O in these two sections:

- ▶ Indexes explained
- ▶ Direct I/O

11.1 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.

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 JPG 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 in the Directory Server Version 6.3 Information Center:

<http://bit.ly/1fg016c>

11.1.1 Optimizing indexes by using DB2 commands

DB2 indexing for Lightweight Directory Access Protocol (LDAP) is evolving. We have learned several ways to improve the indexes that are created. The LDAP directory automatically creates indexes as Example 11-1 shows.

Example 11-1 Creating an index with the LDAP directory

```
db2 CREATE INDEX "LDAPDB2"."LDAP_DESC_AEID" ON "LDAPDB2"."LDAP_DESC" ("DEID"  
ASC,"AEID" ASC) COMPRESS NO ALLOW REVERSE SCANS
```

It is better to create the indexes by using MINPCTUSED 10, as Example 11-2 shows:

Example 11-2 Creating indexes by using MINPCTUSED 10

```
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 MINPCTUSED 10 option dynamically reorganizes the index so that sparsely filled consecutive pages are combined. This minimizes the need to reorganize the index and keeps performance good as directory updates progress. All LDAPDB2 indexes can be optimized this way. You can list all of them by name by using this statement:

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

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

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

However, the uniqueness rule must be used with care. For example, if you want to find all entries with a specific attribute value and there are multiple entries with the same attribute value, do not use this rule, because the resulting index finds only one of the entries.

However, each entry has a unique parent ID, so setting uniqueness on the parent ID index makes sense.

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

11.1.2 Optimizing searches by 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 for both the DB2 instance and database names. This snapshot must be taken while the system is under a full test load 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
```

The resulting file can be large. In earlier sections, we 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. Sample output is shown in Example 11-3.

Example 11-3 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 more than 14,000 seconds for a statement that runs only 163 times. The average execution time is 90 seconds for each search. This is very high. Relatively few physical reads are occurring, but many buffer pool index logical reads are occurring. DB2 is reading much index data in memory, and this is causing high processor consumption and long search times. The snapshot shows the SQL for the troublesome search, but you can figure out what the original LDAP search was.

Notice that LDAP_DESC is used in the SQL. That means it is a subtree search. The only other table that is 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 this form:

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

The snapshot does not show the specific values of DN or UID bases used in the search. This statement might have run many times with different base DN and UID values. However, by looking in the audit log to see whether there is a slow search of this form, there might be an example such as this one:

```
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:

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

2. We create a file called 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 11-4.

Example 11-4 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 dbldplz.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

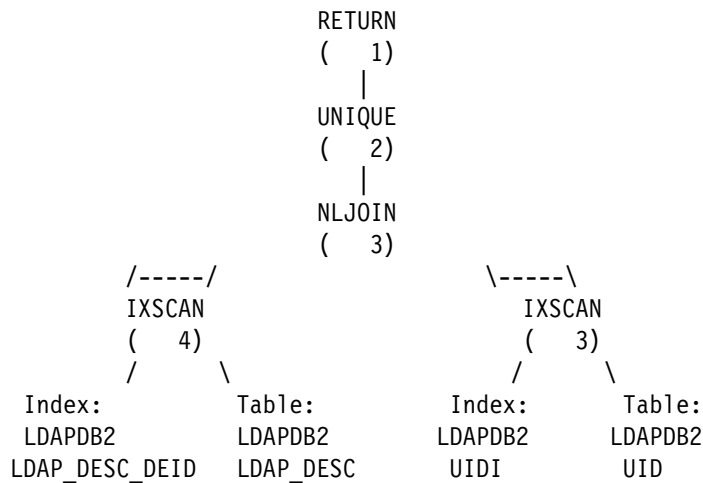
Estimated Cost = 2832063.750000

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:



When executing the query plan, DB2 starts at the lower left. Therefore, it finds all descendants of the particular search base DN. Then, it checks each one to see whether the UID matches (*NLJOIN* stands for *nested loop join*). This is a poor way to do this search, because there are many entries under the specified base DN.

A better way is to find the matching UID first, and then check the descendant table to make sure that the returned entry is a descendant of the base DN. 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 is defined, but it is defined with columns AEID, DEID. It needs to be defined with columns DEID, AEID. Therefore, we fix the index and run **runstats** again. Then we check to see whether the response time for this search is fast, and we get the explanation data again if it is not.

If these steps do not 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. That fixes the query plan for this search. But **LDAP_MAXCARD** can have side effects for other searches, so we must test the effect on all searches to see whether the net effect on the particular workload is positive. An alternative is to set **IBMDSLAPD_USE_SELECTIVITY** to YES, which affects only query plans for large subtrees, by adding a SELECTIVITY clause to the SQL to tell the DB2 optimizer that the LDAP_DESC indexes are going to be less effective than any attribute indexes.

11.2 Direct I/O

By default, on all operating systems for which 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 specified 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 table spaces. All DB2 reads and writes to table spaces are handled as memory mapped I/O within DB2 and are buffered in the DB2 buffer pools. This makes file system buffering redundant, so turn it off.

There are several ways to do this. If raw database-managed storage (DMS) containers for the table spaces are used, I/O is always direct to disk. However, do not use DMS containers for large directories because, within a DMS container, no table can be more than 64 GB. A directory with millions of entries might have tables that are larger than this. Always use system-managed storage (SMS) containers for very large directories. Most of the advantages of DMS containers and direct I/O are realized by disabling file system buffering by running these commands before starting the server:

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

The principal advantage is that processor resource use in reads is reduced by roughly 30%. There is little effect on write loads. If *file system caching* was not turned off when the table space was created, you can turn it off by running the **db2_tunings.sh** script when you are pretuning the DB2 database when you install the LDAP instance. If you created your DB2 instances by DMS and used different names for these table spaces, you must change the names so that they match in this script, too.

For some operating systems, another option is available to boost disk writes. IBM AIX with journaled file system2 (JFS2) supports concurrent I/O. When the NO FILE SYSTEM CACHING option is used on a table space 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 by using the **-cio** option on the **mount** command:

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

This option uses direct I/O implicitly.

On a Linux system, if you are using the Version 2.6 kernel or later, you can enable asynchronous I/O with these steps:

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

```
db2set DB2NOLIOAI0=false
```

You can then restart the database and directory. In general, this approach is more complicated than using NO FILE SYSTEM CACHING; therefore, NO FILE SYSTEM CACHING is preferable.

Disk striping and RAID

By 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 that is 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 is the element size, 64 sectors, times 4 disks, or 256 sectors, which equals 128 KB. The suggested disk striping configuration is shown in Figure 12-1.

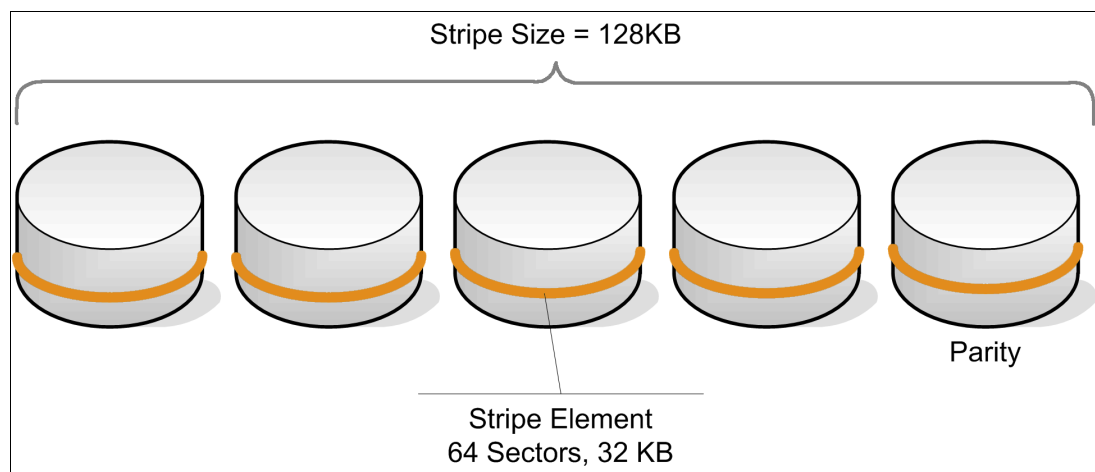


Figure 12-1 Disk striping

12.1 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 high availability and the ability to use volume snapshots and similar means for fast backup and restore of the database.

The preferred practices for RAID use are described in the “Optimizing table space performance when data is on RAID devices” section of the IBM DB2 9.7 Information Center:

<http://ibm.co/liZJWdR>

It includes these key points:

- ▶ Put the DB2 transaction log on a different physical disk than the table spaces.
 - ▶ Either RAID 1 or RAID 5 can be used for table spaces. Use RAID 1 for the transaction log.
 - ▶ A table space might span multiple physical disks, but mount it as a single virtual disk. The RAID system provides the required parallelism. Multiple virtual disks (containers) for the table space slow the throughput by randomizing the I/O. Enable DB2_PARALLEL_IO.
 - ▶ Make the EXTENTSIZE of the table space 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 table space meets these criteria:
 - The RAID stripe size that is multiplied by the number of RAID parallel devices (or a whole multiple of this product)
 - And is
 - 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. Make sure that each busy database has its own set of physical disks. For example, move the transaction logs to another mount point so that they are not on the same disk as the database. By default, they are on the same disk space.

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



Distributed directory

This chapter explains the necessity of performance tuning and relevant aspects in a distributed directory environment. It also describes the proxy server feature. It includes the following sections:

- ▶ Distributed directory environment
- ▶ Strategies for partitioning data
- ▶ Fail over and load balancing
- ▶ Tuning the proxy server

13.1 Distributed directory environment

In this first section, we review the following details:

- ▶ The need for distributed directories
- ▶ Distributed directory and proxy server
- ▶ Terminology related to a distributed directory and proxy server
- ▶ Configuring a distributed directory environment
- ▶ A closer look at proxy server configuration

13.1.1 The need for distributed directories

IBM Directory Server scalability and performance requirements have changed over time. The Directory Server is now expected to store hundreds of millions of entries and, at the same time, provide very high performance. Storing such large amounts of data in a single Directory Server is likely to affect its performance and, more importantly, reaches the limits of scalability issues for the operating system and hardware. Distributed directories were introduced to address these performance and scalability issues.

13.1.2 Distributed directory and proxy server

In a distributed directory environment, the data is partitioned across multiple Directory Servers. In such a distributed directory, a group of systems or instances are involved, including the normal Directory Servers and proxy servers. A normal Directory Server instance (a Lightweight Directory Access Protocol, or LDAP, *instance*) with a database (relational database management system, or RDBM) holds the data. These normal Directory Server instances, also called *containers*, act as a back end to the proxy server instances. The proxy server instances act as front ends of the distributed directory. A proxy server sits in front of multiple Directory Servers and provides a layer of abstraction between the set of back-end directories and the client applications. The proxy server does not store any data because it does not have a dedicated database.

In a distributed directory environment with proxy servers, the client applications interact with the proxy server. This type of distributed directory environment provides a unified directory view to the client applications, even though the actual data is distributed among the back-end containers. The proxy server provides request routing, load balancing, and failover functions. It also supports distributed authentication, as well as distributed groups and memberships.

13.1.3 Terminology related to a distributed directory and proxy server

This section lists commonly used terms in the context of the Directory Server proxy server. Relevant configuration attributes are written within parenthesis where applicable.

Split

A specific namespace is partitioned into a set of partitions, each of which is in an independent Directory Server instance. Each of these partitions is referred to as a *split*. The namespace that is being split is referred by **ibm-slapdProxyPartitionBase**, although the number of partitions for a specific namespace is specified by **ibm-slapdProxyNumPartitions**.

Partition Index (`ibm-slapdProxyPartitionIndex`)

Each partition or split for a specific namespace is represented by an index that is known as the *partition index*. When resolving a specific entry location, the proxy partitioning algorithm returns a hash value, which is then used to resolve the corresponding back-end server. This hash value can be regarded as an alias for a partition index.

ServerGroup

For a particular partition, replication is normally set up across the set of servers that are serving that partition. If one of the servers in the partition is running, the proxy marks the partition as active. A ServerGroup is a way of specifying a set of servers where, if any of the servers are up, the proxy can mark the relevant partition as active even if the rest of the servers in the group are down.

Global Administrative Group members

Global Administrative Group members are users who have the administrative privileges for accessing entries in the back-end server. However, they have no privileges or access rights to any data or operations that are related to the configuration of the back-end Directory Server. All Global Administrative Group members have the same set of privileges. For a directory administrator, the Global Administrative Group is a way to delegate administrative rights to the back-end directory data in a distributed environment.

Local Administrative Group members

Local Administrative Group members are users who have a subset of administrative privileges. All the Local Administrative Group members have the same set of privileges. For a directory administrator, this group is a way to delegate a limited set of administrative tasks to one or more individual user accounts. These users can perform most of the administrative tasks. Exceptions to these tasks are operations that might increase the privileges of those users, such as changing the password of the directory administrator or clearing the audit log.

Connection Pool Size (`ibm-slapdProxyConnectionPoolSize`)

Each proxy can be configured to communicate with each of the back-end servers over a set of connections. These connections are in the form of a pool, where all of the connections are established when the proxy starts and are used when required. This parameter is configurable and can be different for different back-end servers.

Proxy DN (`ibm-slapdProxyDN`)

This is the distinguished name (DN) that a proxy server binds to the back-end servers. Basically, this DN proxies the user binding to the proxy server.

Proxy Target URL (`ibm-slapdProxyTargetURL`)

This attribute is used by the proxy server to specify the URL of the back-end server.

Note: The user binding to the proxy server is considered an administrator if it is the directory administrator, or a Global Administrative Group member, or a Local Administrative Group member.

13.1.4 Configuring a distributed directory environment

This section mentions the high-level steps that are involved in distributed directory environment. For detailed configuration steps, such as commands or GUI methods, see the Administration guide in the online information center:

http://pic.dhe.ibm.com/infocenter/tivihelp/v2r1/topic/com.ibm.IBMDS.doc_6.0/admin_gd19.htm

1. Configure the proxy server. You can create the initial instance through either the **idsicrt** command-line tool or the **idsxinst** instance administration tool.
2. Start the proxy server in configuration mode. Then proceed with the main configuration of proxy server to configure the partition bases, back-end servers, and partitions.

Steps to set up this scenario

The following guidance is for the high-level steps that are involved in setting up the relative distinguished name (IBM RDN®) hash-based scenario that is described in this section. The steps are based on the assumption that the necessary product set is already installed on the system.

1. Plan your distributed directory environment, and decide what data to keep on which back-end servers
2. Configure the back-end servers similar to normal Directory Servers:
 - a. Create the Directory Server instance (which also creates the associated IBM DB2 instance).
 - b. Create and configure the database.
 - c. Configure the administrator DN and password.
 - d. Configure necessary suffixes.
 - e. Apply necessary schema and configuration updates.

Note: Do not start the directory servers yet.

3. Create the instance and configure the proxy server.
 - a. Create the proxy server instance.
 - b. Configure the administrator DN and password.
 - c. Start the proxy in configuration-only mode.
 - d. Configure the partition base (suffix).
 - e. Configure back-end server entries.
 - f. Configure partitions and splits.
 - g. If there were schema changes, copy the schema files from the back-end servers into the etc folder of the proxy server instance.
 - h. Stop the proxy server in configuration-only mode.

4. Split and load the data:

Note: If you have single, large data in an LDAP Directory Interchange Format (LDIF) file, you must split the data before loading it into the back-end servers. A tool called *ddsetup* is used to split the data in case if your directory uses RDN hash-based splitting.

- a. Create a new *ddsetup* configuration file that is based on the proxy configuration
- b. Using this *ddsetup* configuration file and LDIF data file as inputs to the **ddsetup** tool, split the data into multiple files. There are separate LDIF files for each back-end server. If there are sub-based splits, the data can also be split directly without using **ddsetup**.

But for RDN-based splits, it is best to use **ddsetup**, because it uses the same RDN hash plug-in that is used by the proxy server to split the data.

- c. Load the individual LDIF files onto back-end servers by using bulk load (especially if there are large LDIF files) or **ldif2db**.
5. Tune the back-end servers. Overall distributed directory performance depends directly on the back-end server performance to a maximum extent. It is very important to tune the back-end servers appropriately and on regular schedules.
6. Start the back-end servers.
7. Start the proxy server in normal mode.

13.1.5 A closer look at proxy server configuration

Now, examine various sections of the proxy server configuration, which are mainly in the `ibmslapd.conf` file of the proxy server instance's `etc` folder.

Proxy configuration

The base entry of `cn=Configuration` contains an attribute called `ibm-slapdServerBackend`, which should contain a value of `PROXY`. This means that the instance is now configured for a proxy back end rather than an RDBM back end, as Example 13-1 shows.

Example 13-1 Proxy configuration

```
dn: cn=Configuration
...
ibm-slapdServerBackend: PROXY
...
```

Proxy Backends container

This is just a top-level container within the configuration for all of the proxy-related configuration, as shown in Example 13-2.

Example 13-2 Proxy backends container

```
dn: cn=Proxy Backends, cn=IBM Directory, cn=Schemas, cn=Configuration
cn: Proxy Backends
objectclass: top
objectclass: container
```

Proxy Backend database entry

The Proxy Backend database entry is an important part of configuring the proxy server. It defines the plug-ins that are configured in the proxy server instance and includes the plug-in for DN partitioning (which provides the RDN hash function).

In a distributed group, group entries and member DNs are in different partitions. *Distributed dynamic groups* are dynamic groups that are defined when some or all of the members are in a different partition. The Proxy Backend database entry describes whether the distributed directory environment contains the distributed groups or distributed dynamic groups.

This Proxy Backend database entry includes the partition bases (also called the proxy instance's *suffixes*). The attributes that control paged search support are also included in this entry. Example 13-3 on page 96 shows a typical Proxy Backend entry. The tunable parameters on this entry are described in 13.4.2, "Tuning attributes in Proxy Backend database entry:" on page 128.

Example 13-3 Proxy Backend database entry

```
dn: cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas, cn=Configuration
cn: Proxy Backend
ibm-slapdDNPartitionPlugin: libldapdnhash.<extn> dnHashInit
ibm-slapdPagedResAllowNonAdmin: TRUE
ibm-slapdPagedResLmt: 3
ibm-slapdPlugin: database libback-proxy.<extn> proxy_backend_init
ibm-slapdPlugin: extendedop libback-proxy.s<extn> initResumeRole
ibm-slapdProxyEnableDistDynamicGroups: true
ibm-slapdProxyEnableDistGroups: true
ibm-slapdSuffix: dc=com
ibm-slapdSuffix: cn=ibmpolicies
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackend
```

Note: In that example, the plug-in extension <extn> is platform-dependent and can contain values such as .dll, .a, and .so.

Back-end server

A back-end server entry in proxy configuration is under the Proxy Backend database. At least one back-end server is required in a proxy configuration. Depending on the distributed directory infrastructure, there might be more than one server present in a proxy back end, which means that the proxy recognizes those back-end servers to connect and bind.

Example 13-4 shows a typical back-end server entry. These are the important attributes in the section:

- ▶ **ibm-slapdProxyTargetURL** (contains a valid LDAP URL for the back-end server)
- ▶ **ibm-slapdProxyDN** (the bind DN)
- ▶ **ibm-slapdProxyPW** (the bind password)

The proxy server uses these parameters to establish a connection with the back-end server.

Example 13-4 Backend server entry

```
dn: cn=Server1, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server1
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckLimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backend1:389
ibm-slapdServerID: 395e8a82-57f8-448c-bb59-953aa8ad6143
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry
```

The other tunable attributes in the back-end server entry are covered later in 13.4.3, “Tuning attributes in Backend server entry” on page 130.

Server groups

Server groups in a proxy configuration inform the proxy that the configured list of back-end servers within that group contain data. The proxy balances the load and handles the failover if those server group entries are configured. It is optional to configure a server group entry in a proxy configuration. It is required if the back ends contain replicated data and you want to take advantage of load balancing and failover. Example 13-5 shows a typical serverGroup entry in proxy configuration.

Example 13-5 Server group entry

```
dn: cn=serverGroup, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: serverGroup
ibm-slapdProxyBackendServerDN: cn=Server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerDN: cn=Server2,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendServerGroup
```

Partition base (Backend Split Container)

A partition base entry (also referred to as a backend split container) is under the proxy DB entry in configuration. This entry defines the subtree on which the partition base (**ibm-slapdProxyPartitionBase**) is defined and number of partitions or splits (**ibm-slapdProxyNumPartitions**) under this base. Example 13-6 shows a typical partition base entry. The tunable parameters on this entry are covered in 13.4.4, “Tuning attributes in partition base entry” on page 131.

Example 13-6 Partition base entry

```
dn: cn=dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: dc=com split
ibm-slapdProxyAutoFailBack: true
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 3
ibm-slapdProxyPartitionBase: dc=com
ibm-slapdProxySplitName: dccomsplit
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
```

Partitions (Backend Split)

Under a Partition base entry, one or more Partitions, or *splits*, are defined. A Partition base entry is also be referred as a *Backend Split* entry. At least one partition (split) entry is required for each partition base. This entry informs the proxy that a split of a partition base is located on a particular server (**ibm-slapdProxyBackendServerDN**) with a particular index (**ibm-slapdProxyPartitionIndex**) and role (**ibm-slapdProxyBackendServerRole**). A typical partition (split) is shown in Example 13-7 on page 98. The tunable parameters on this entry are described later in 13.4.5, “Tuning attributes in partition entry” on page 132.

Example 13-7 Partition split

```
dn: cn=split1, cn=dc=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: split1
ibm-slapdProxyBackendServerDN: cn=Server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit
```

13.2 Strategies for partitioning data

There are different ways of partitioning (splitting) data in a distributed directory environment. The currently available mechanisms are RDN hash-based splitting and subtree-based splitting. Both RDN hash and subtree splitting can be combined to form customized data partitions.

13.2.1 RDN hash-based splitting

In RDN hash-based splitting, a hash that is based on these three factors is generated for a specific entry:

- ▶ Parent entry
- ▶ Partition base
- ▶ Number of partitions

This hash value is mapped to a specific back-end directory server. This kind of split can lead to uneven data splits, so different splits might contain a different number of entries.

If the directory information tree (DIT) is breadth-centric and has entries that cannot be placed under discrete partition bases (subtrees), the RDN hash-based distributed directory setup is more beneficial.

Scenario for RDN hash-based splitting

In this scenario, the proxy server is configured with two partition bases:

- ▶ cn=ibmpolicies
- ▶ dc=com

The dc=com data is split into two partitions in two back-end servers by using RDN hash. The cn=ibmpolicies data is maintained in one partition on two back-end servers, which means all of the cn=ibmpolicies data is present on each of the back-end servers. It is also preferable to have the cn=ibmpolicies configured for replication between back-end servers to maintain the same set of policies and the same schema.

Suggestion: It is best to keep cn=ibmpolicies the same across all back-end servers. When it is configured for replication among the back-end servers, the live LDAP modifications on cn=schema and cn=ibmpolicies changes are synchronized among the back-end servers automatically.

This RDN hash-based scenario is represented in the workflow diagram in Figure 13-1.

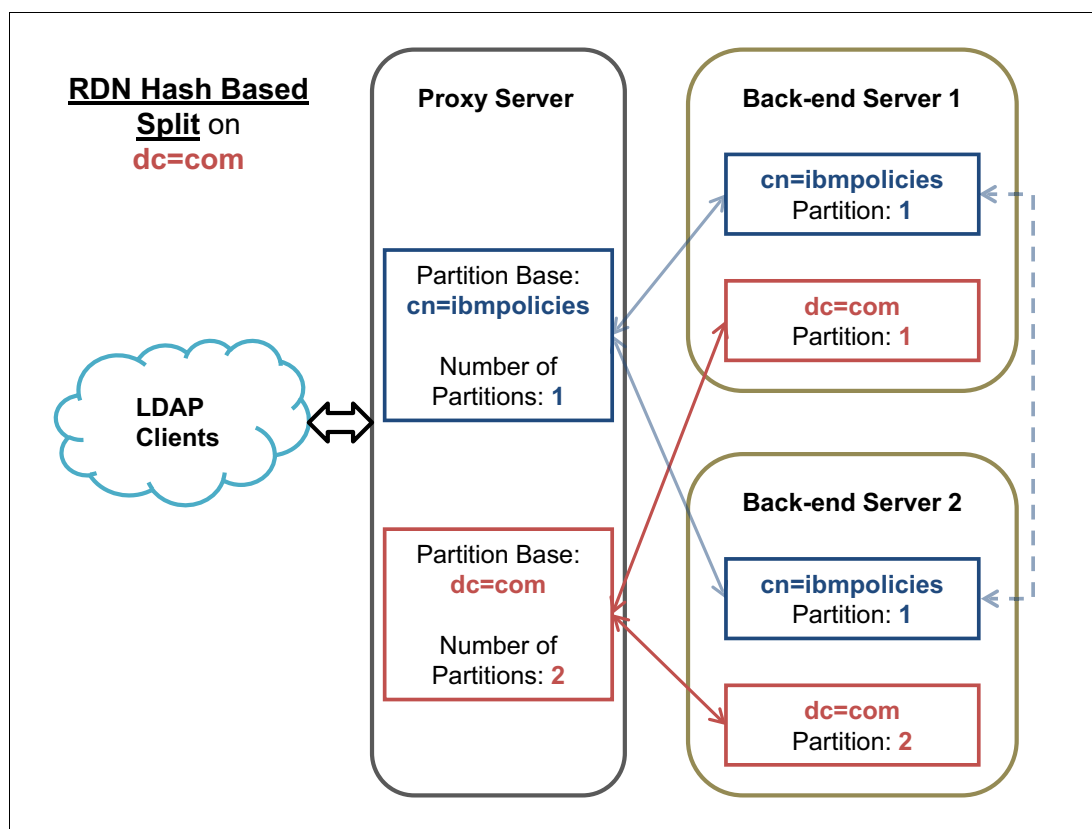


Figure 13-1 RDN hash-based scenario

LDAP requests from clients that are coming into the proxy server on the `dc=com` suffix are being routed to the back-end servers based on the nature of the request.

All update operations are routed to individual back-end servers based on the RDN hash of the entry that is being updated.

For search requests made on the `dc=com` base with subtree scope, the proxy gets the results from both back-end servers and provides the unified results to the LDAP client. However, if search requests are made on any specific subtree or entry under `dc=com`, the proxy server uses the RDN hash method to send the search request to only a specific back-end server.

If a search request is made on the `cn=ibmpolicies` branch, the request is routed to any one of the back-end servers. Because the same data is maintained in a single partition, getting search results from any one back-end server is sufficient.

Note: Proxy servers do not support null-based search for subtree scope to get data from back-end servers. You must search one of the configured partition bases (suffixes) of the proxy server to get the data from the back-end servers.

Configuration details

Now, examine the configuration of this scenario closely in terms of LDIF snippets from the proxy LDAP server instance's `ibmslapd.conf` file.

Example 13-8 on page 100 shows the proxy configuration's back-end entry. The `dc=com` and `cn=ibmpolicies` suffixes are configured in this entry through `ibm-slapdSuffix`.

Example 13-8 Proxy configuration's back-end entry

```
dn: cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas, cn=Configuration
cn: Proxy Backend
ibm-slapdDNPartitionPlugin: libldapdnhash.<extn> dnHashInit
ibm-slapdPagedResAllowNonAdmin: TRUE
ibm-slapdPagedResLmt: 3
ibm-slapdPlugin: database libback-proxy.<extn> proxy_backend_init
ibm-slapdPlugin: extendedop libback-proxy.<extn> initResumeRole
ibm-slapdProxyEnableDistDynamicGroups: true
ibm-slapdProxyEnableDistGroups: true
ibm-slapdSuffix: dc=com
ibm-slapdSuffix: cn=ibmpolicies
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackend
```

The proxy configuration's back-end server entries are shown in Example 13-9. From these server entries, the proxy connects to the back-end server that was configured through **ibm-slapdProxyTargetURL** by using the bind DN specified in **ibm-slapdProxyDN** and its password in **ibm-slapdProxyPW**.

Example 13-9 Proxy configuration's back-end server entries

```
dn: cn=Server1, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server1
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckOlimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backend1:389
ibm-slapdServerID: 395e8a82-57f8-448c-bb59-953aa8ad6143
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry
```

```
dn: cn=Server2, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server2
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckOlimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backend2:389
ibm-slapdServerID: 014ac3eb-1aa2-4682-9cd3-4dbd2548bd44
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry
```

The proxy configuration's **cn=ibmpolicies** partition definitions are shown in Example 13-10. The partition base (**ibm-slapdProxyPartitionBase**) is configured in **cn=ibmpolicies** with a single partition (**ibm-slapdProxyNumPartitions: 1**). The remaining two entries define that Partition 1 exists on both the back-end servers and, importantly, that the partition index (**ibm-slapdProxyPartitionIndex: 1**) is the same on both back-end servers.

Example 13-10 Proxy configuration's cn=ibmpolicies partition definitions

```
dn: cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: cn=ibmpolicies split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 1
ibm-slapdProxyPartitionBase: cn=ibmpolicies
ibm-slapdProxySplitName: ibmpolicies
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
objectclass: top

dn: cn=split1, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split1
ibm-slapdProxyBackendServerDN: cn=Server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

dn: cn=split2, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split2
ibm-slapdProxyBackendServerDN: cn=Server2,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit
```

The proxy configuration's **dc=com** partition definitions are shown in Example 13-11 on page 102. The partition base (**ibm-slapdProxyPartitionBase**) is configured on **dc=com** with a two partitions (**ibm-slapdProxyNumPartitions: 2**).

The remaining two entries define that Partition 1 exists on the first back-end server with the partition index (**ibm-slapdProxyPartitionIndex**) value of 1, and Partition 2 exists on the second back-end server with the partition index value of 2.

Example 13-11 Proxy configuration's dc=com partition definitions

```
dn: cn=dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: dc=com split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 2
ibm-slapdProxyPartitionBase: dc=com
ibm-slapdProxySplitName: dccom
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
objectclass: top
```

```
dn: cn=split1, cn=dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: split1
ibm-slapdProxyBackendServerDN: cn=Server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit
```

```
dn: cn=split2, cn=dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: split2
ibm-slapdProxyBackendServerDN: cn=Server2,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 2
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit
```

13.2.2 Subtree-based splitting

In the subtree-based split, the back-end servers are used as containers for an entire subtree. Therefore, if the data is scattered across a set of subtrees, each subtree can be mapped to a different directory server.

Using subtree-based split is more suitable than an RDN hash-based split when you want uniform data distribution. The subtree where the split is might be a suffix or a branch under suffix or even branch of another branch.

Advantages of subtree splitting include ease of setup, better capacity planning, feasibility of data redistribution, and scalability.

Scenario for subtree-based splitting

In this scenario, the proxy server is configured with three partition bases:

- ▶ cn=ibmpolicies
- ▶ ou=dept1,dc=com
- ▶ ou=dept2,dc=com

The data that corresponds to **cn=ibmpolicies** is maintained in one partition on both back-end servers, which means that all of the cn=ibmpolicies data is present on each of the back-end servers. Its also preferable to have the cn=ibmpolicies configured for replication between back-end servers to maintain same set of policies and schema. The ou=dept1,dc=com data is kept only on the first back-end server, whereas ou=dept2,dc=com data is kept only on the second back-end server.

This subtree-based scenario is represented in Figure 13-2.

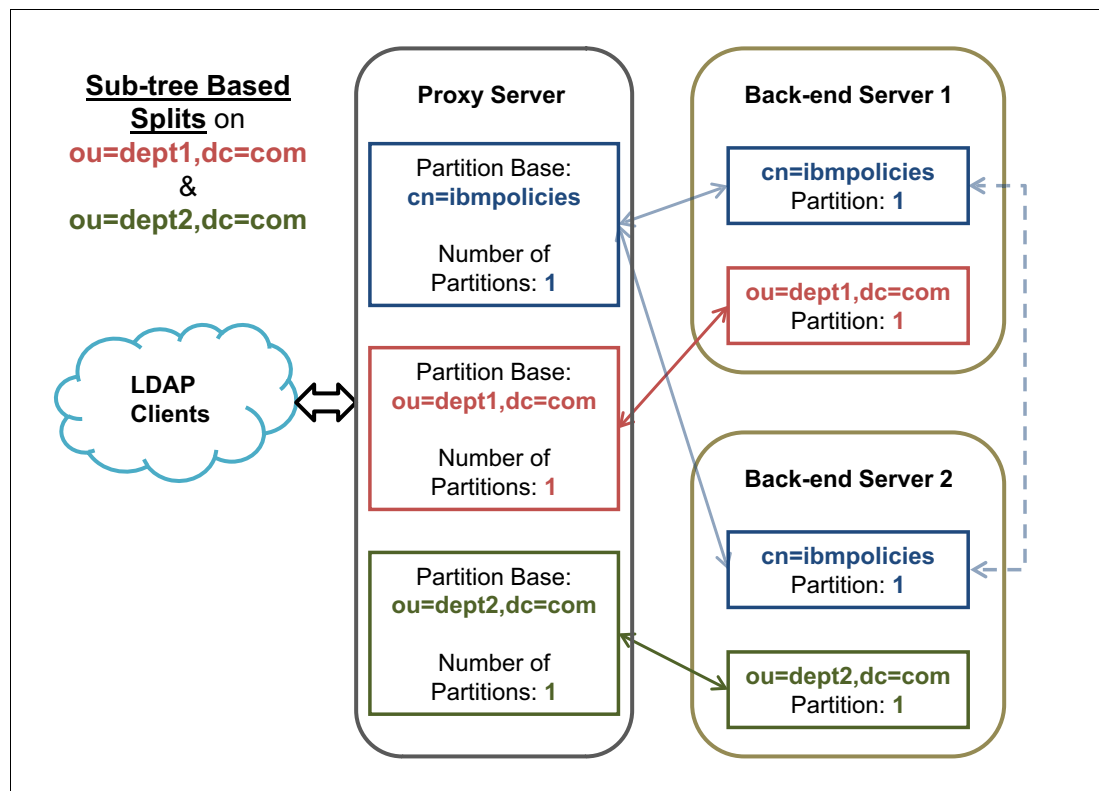


Figure 13-2 Subtree-based scenario

LDAP requests from clients that are coming into the proxy server on ou=dept1,dc=com suffix are routed to the back-end server1 and the requests on the ou=dept2,dc=com suffix are routed to back-end server2.

If a search request is made on the cn=ibmpolicies branch, the request is routed to any one of the back-end servers. The same data is maintained in a single partition, so getting search results from any one back-end server is sufficient.

Configuration details

Examine the configuration of this scenario closely in terms of LDIF snippets from the proxy LDAP server instance's ibmslapd.conf file.

Example 13-12 on page 104 shows the proxy configuration's back-end entry. These three suffixes are configured in this entry through **ibm-slapdSuffix**:

- ▶ ou=dept1,dc=com
- ▶ ou=dept2,dc=com
- ▶ cn=ibmpolicies

Example 13-12 Proxy configuration's back-end entry

```
dn: cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas, cn=Configuration
cn: Proxy Backend
ibm-slapdDNPartitionPlugin: libldapdnhash.<extn> dnHashInit
ibm-slapdPagedResAllowNonAdmin: TRUE
ibm-slapdPagedResLmt: 3
ibm-slapdPlugin: database libback-proxy.<extn> proxy_backend_init
ibm-slapdPlugin: extendedop libback-proxy.<extn> initResumeRole
ibm-slapdProxyEnableDistDynamicGroups: true
ibm-slapdProxyEnableDistGroups: true
ibm-slapdSuffix: ou=dept1,dc=com
ibm-slapdSuffix: ou=dept2,dc=com
ibm-slapdSuffix: cn=ibmpolicies
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackend
```

The proxy configuration's back-end server's entries are shown in Example 13-13. The proxy connects to the back-end server that is configured through **ibm-slapdProxyTargetURL** by using the bind DN specified in **ibm-slapdProxyDN** and its password in **ibm-slapdProxyPW**.

Example 13-13 Proxy configuration's back-end server's entries

```
dn: cn=Server1, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server1
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckOlimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backend1:389
ibm-slapdServerID: 395e8a82-57f8-448c-bb59-953aa8ad6143
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry

dn: cn=Server2, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server2
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckOlimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backend2:389
ibm-slapdServerID: 014ac3eb-1aa2-4682-9cd3-4dbd2548bd44
ibm-slapdStatusInterval: 0
```



```
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry
```

The proxy configuration's **cn=ibmpolicies** partition definitions are shown in Example 13-14. From the first entry, the partition base (**ibm-slapdProxyPartitionBase**) is configured on **cn=ibmpolicies** with a single partition (**ibm-slapdProxyNumPartitions: 1**). The remaining two entries define that Partition 1 exists on both the back-end servers and that the partition index (**ibm-slapdProxyPartitionIndex: 1**) is the same on both back-end servers.

Example 13-14 Proxy configuration's cn=ibmpolicies partition definitions

```
dn: cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: cn=ibmpolicies split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 1
ibm-slapdProxyPartitionBase: cn=ibmpolicies
ibm-slapdProxySplitName: ibmpolicies
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
objectclass: top
```

```
dn: cn=split1, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split1
ibm-slapdProxyBackendServerDN: cn=Server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit
```

```
dn: cn=split2, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split2
ibm-slapdProxyBackendServerDN: cn=Server2,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit
```

The proxy configuration's **ou=dept1,dc=com** partition and **ou=dept2,dc=com** definitions are shown in Example 13-15 on page 106. From the first and third entries, the partition base values are configured on **ou=dept1,dc=com** and **ou=dept2,dc=com**, respectively, each with a single partition. The remaining two entries (second and fourth) define that the partition of

ou=dept1,dc=com exists on the first back-end server and partition of ou=dept2,dc=com exists on the second back-end server.

Example 13-15 Proxy configuration's ou=dept1,dc=com partition and ou=dept2,dc=com definitions

```
dn: cn=ou\=dept1\,dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: ou=dept1,dc=com split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 1
ibm-slapdProxyPartitionBase: ou=dept1,dc=com
ibm-slapdProxySplitName: dept1split
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
objectclass: top
```

```
dn: cn=split1, cn=ou\=dept1\,dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split1
ibm-slapdProxyBackendServerDN: cn=Server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit
```

```
dn: cn=ou\=dept2\,dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: ou=dept2,dc=com split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 1
ibm-slapdProxyPartitionBase: ou=dept2,dc=com
ibm-slapdProxySplitName: dept2split
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
objectclass: top
```

```
dn: cn=split2, cn=ou\=dept2\,dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split2
ibm-slapdProxyBackendServerDN: cn=Server2,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 2
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
```

13.2.3 Hybrid splits

In a hybrid split configuration of proxy server, the subtree-based and the RDN hash-based solutions are combined to map some subtrees to a single, unique back end and to partition the rest of the subtrees based on the RDN hash. Depth-centric subtrees are kept in dedicated back ends by using subtree splits. The remaining data is spread across based on RDN hash.

Scenario for hybrid split

In this scenario, the proxy server is configured with three partition bases:

- ▶ `cn=ibmpolicies`
- ▶ `ou=dept1,dc=com`
- ▶ `ou=dept2,dc=com`

The data that corresponds to `cn=ibmpolicies` is maintained in one partition on all back-end servers, which means all the `cn=ibmpolicies` data is present on each of the back-end servers. It is also preferable to have the `cn=ibmpolicies` configured for replication between back-end servers to maintain same set of policies and schema. The `ou=dept1,dc=com` data is spread across the first and second back-end servers, but `ou=dept2,dc=com` data is kept only on the third back-end server. This hybrid scenario is represented in the workflow diagram in Figure 13-3.

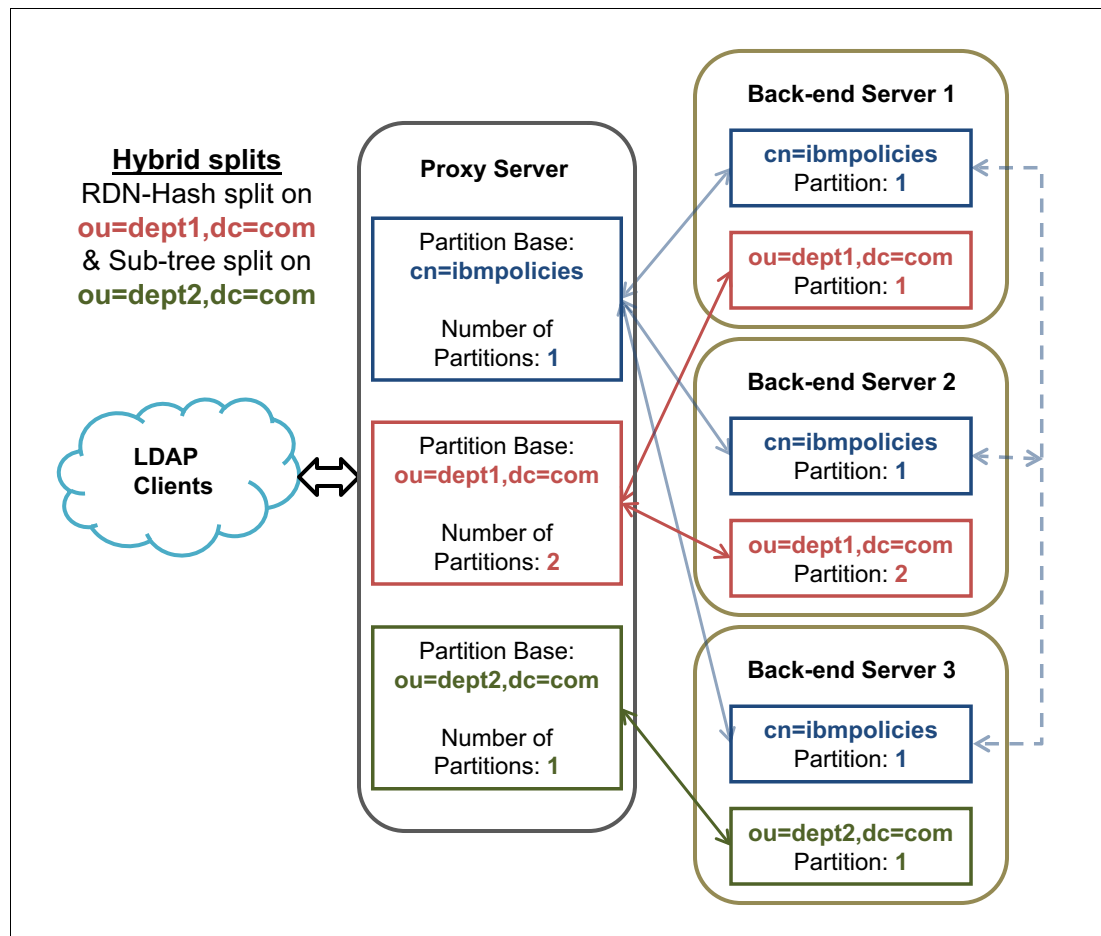


Figure 13-3 Hybrid split scenario

LDAP requests that are coming into the proxy server on ou=dept1,dc=com suffix are routed to the back-end Server 1 and back-end Server 2, based on the entry's RDN hash, and the requests on the ou=dept2,dc=com suffix are routed to back-end Server 3.

If a search request is made cn=ibmpolicies branch, the request is routed to any one of the back-end servers only. Since the same data is maintained in a single partition, getting search results from any one back-end server is sufficient.

Configuration details

Examine the configuration of this scenario closely in terms of snippets of LDIF from the proxy LDAP server instance's `ibmslapd.conf` file.

Proxy configuration's back-end entry is seen in Example 13-16. Three suffixes ou=dept1,dc=com, ou=dept2,dc=com and cn=ibmpolicies are configured in this entry through **ibm-slapdSuffix**.

Example 13-16 Proxy configuration's back-end entry

```
dn: cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas, cn=Configuration
cn: Proxy Backend
ibm-slapdDNPartitionPlugin: libldapdnhash.<extn> dnHashInit
ibm-slapdPagedResAllowNonAdmin: TRUE
ibm-slapdPagedResLmt: 3
ibm-slapdPlugin: database libback-proxy.<extn> proxy_backend_init
ibm-slapdPlugin: extendedop libback-proxy.<extn> initResumeRole
ibm-slapdProxyEnableDistDynamicGroups: true
ibm-slapdProxyEnableDistGroups: true
ibm-slapdSuffix: ou=dept1,dc=com
ibm-slapdSuffix: ou=dept2,dc=com
ibm-slapdSuffix: cn=ibmpolicies
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackend
```

The proxy configuration back-end server's entries are shown in Example 13-17. The proxy connects to the back-end server configured through **ibm-slapdProxyTargetURL** by using the bind DN specified in **ibm-slapdProxyDN** and its password in **ibm-slapdProxyPW**.

Example 13-17 Proxy configuration's back-end server's entries

```
dn: cn=Server1, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server1
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckOlimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backend1:389
ibm-slapdServerID: 395e8a82-57f8-448c-bb59-953aa8ad6143
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry
```

```

dn: cn=Server2, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server2
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckOlimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backend2:389
ibm-slapdServerID: 014ac3eb-1aa2-4682-9cd3-4dbd2548bd44
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry

```

```

dn: cn=Server3, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server3
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckOlimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backend3:389
ibm-slapdServerID: ac308bc9-c347-4ed4-a14b-df1c272d378e
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry

```

The proxy configuration's **cn=ibmpolicies** partition definitions are shown in Example 13-18. From the first entry, the partition base (`ibm-slapdProxyPartitionBase`) is configured on `cn=ibmpolicies` with a single partition (`ibm-slapdProxyNumPartitions: 1`). The remaining two entries define that the partition 1 exists on both the back-end servers and importantly the partition index (`ibm-slapdProxyPartitionIndex: 1`) is the same on both back-end servers.

Example 13-18 Proxy configuration's `cn=ibmpolicies` partition definitions

```

dn: cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: cn=ibmpolicies split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 1
ibm-slapdProxyPartitionBase: cn=ibmpolicies
ibm-slapdProxySplitName: ibmpolicies
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
objectclass: top

```

```

dn: cn=split1, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration

```

```

cn: split1
ibm-slapdProxyBackendServerDN: cn=Server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

dn: cn=split2, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split2
ibm-slapdProxyBackendServerDN: cn=Server2,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

dn: cn=split3, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split3
ibm-slapdProxyBackendServerDN: cn=Server3,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

```

The proxy configuration's ou=dept1,dc=com partition definitions are shown in Example 13-19. From the first entry, the partition base is configured on ou=dept1,dc=com with two partitions. The remaining two entries (second and third) define that the partition of ou=dept1,dc=com is split into on the first and second back-end servers by using an RDN hash split.

Example 13-19 Proxy configuration's ou=dept1,dc=com partition definitions

```

dn: cn=ou\=dept1\,dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: ou=dept1,dc=com split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 2
ibm-slapdProxyPartitionBase: ou=dept1,dc=com
ibm-slapdProxySplitName: dept1split
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
objectclass: top

```

```

dn: cn=split1, cn=ou\=dept1\,dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split1
ibm-slapdProxyBackendServerDN: cn=Server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

dn: cn=split2, cn=ou\=dept1\,dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split2
ibm-slapdProxyBackendServerDN: cn=Server2,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 2
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

```

The proxy configuration's ou=dept2,dc=com partition definitions are shown in Example 13-20. From the first entry, the partition base is configured on ou=dept2,dc=com with one partition. The second entry defines that the partition of ou=dept2,dc=com is available on the third back-end server as a complete subtree.

Example 13-20 Proxy configuration's ou=dept2,dc=com partition definitions

```

dn: cn=ou\=dept2\,dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: ou=dept2,dc=com split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 2
ibm-slapdProxyPartitionBase: ou=dept2,dc=com
ibm-slapdProxySplitName: dept2split
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
objectclass: top

dn: cn=split1, cn=ou\=dept2\,dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split1
ibm-slapdProxyBackendServerDN: cn=Server3,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry

```

13.2.4 Nested splits

In nested splits, the top-level DIT is distributed based on the RDN hash, and specific subtrees within the same DIT are mapped to dedicated back-end servers.

Scenario for nested splits

In this scenario, the proxy server is configured with four partition bases:

- ▶ cn=ibmpolicies
- ▶ dc=com
- ▶ ou=dept1,dc=com
- ▶ ou=dept2,dc=com

The data that corresponds to cn=ibmpolicies is maintained in one partition on all back-end servers. This means that all of the cn=ibmpolicies data is present on each of the back-end servers. It is also preferable to have the cn=ibmpolicies configured for replication between back-end servers to maintain same set of policies and schema. The dc=com is split based an RDN hash across back-end Servers 1 and 2. The ou=dept1,dc=com data is kept only on the third back-end server, and the ou=dept2,dc=com data is kept only on the fourth back-end server. This nested scenario is represented in Figure 13-4 on page 113.

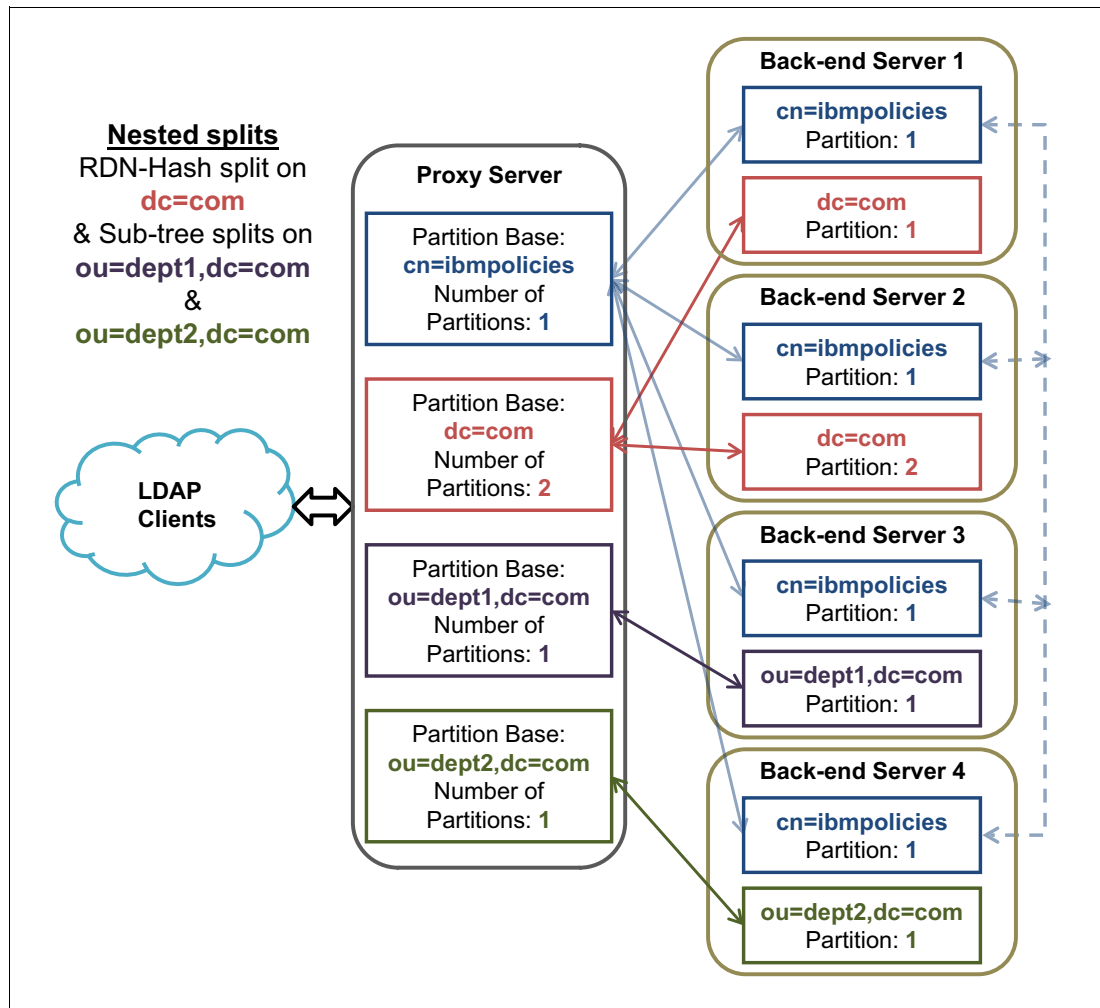


Figure 13-4 Nested split scenario

LDAP requests from clients coming into the proxy server on **ou=dept1,dc=com** suffix are routed to the back-end Server 3 and the requests on the **ou=dept2,dc=com** suffix are routed to back-end server 4. All other requests on **dc=com** are routed to either back-end Server 1 or 2 based on the RDN hash.

If a search request is made **cn=ibmpolicies** branch, the request is routed to any one of the back-end servers only. Since the same data is maintained in a single partition, getting search results from any one back-end server is sufficient.

Configuration details

Examine the configuration of this scenario closely in terms of snippets of LDIF from the proxy LDAP server instance's **ibmslapd.conf** file.

Proxy configuration's back-end entry is seen in Example 13-21. Two suffixes, **dc=com** and **cn=ibmpolicies**, are configured in this entry through **ibm-slapdSuffix**.

Example 13-21 Proxy configuration's back-end entry

```
dn: cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas, cn=Configuration
cn: Proxy Backend
ibm-slapdDNPartitionPlugin: libldapdnhash.<extn> dnHashInit
ibm-slapdPagedResAllowNonAdmin: TRUE
```

```
ibm-slapdPagedResLmt: 3
ibm-slapdPlugin: database libback-proxy.<extn> proxy_backend_init
ibm-slapdPlugin: extendedop libback-proxy.<extn> initResumeRole
ibm-slapdProxyEnableDistDynamicGroups: true
ibm-slapdProxyEnableDistGroups: true
ibm-slapdSuffix: dc=com
ibm-slapdSuffix: cn=ibmpolicies
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackend
```

The proxy configuration's back-end server entries are shown in Example 13-22. The proxy connects to the back-end server configured through **ibm-slapdProxyTargetURL** by using the bind DN specified in **ibm-slapdProxyDN** and its password in **ibm-slapdProxyPW**.

Example 13-22 Proxy configuration's back-end server entries

```
dn: cn=Server1, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server1
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckOlimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backend1:389
ibm-slapdServerID: 395e8a82-57f8-448c-bb59-953aa8ad6143
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry
```

```
dn: cn=Server2, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server2
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckOlimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backend2:389
ibm-slapdServerID: 014ac3eb-1aa2-4682-9cd3-4dbd2548bd44
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry
```

```
dn: cn=Server3, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server3
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckOlimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
```

```

ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backend3:389
ibm-slapdServerID: ac308bc9-c347-4ed4-a14b-df1c272d378e
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry

dn: cn=Server4, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server4
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckOlimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backend4:389
ibm-slapdServerID: dd5cdd34-ff59-4a9d-86a0-ce063224cabf
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry

```

The proxy configuration's `cn=ibmpolicies` partition definitions are shown in Example 13-23. From the first entry, the partition base (**`ibm-slapdProxyPartitionBase`**) is configured on `cn=ibmpolicies` with a single partition (**`ibm-slapdProxyNumPartitions: 1`**). The remaining four entries define that the Partition 1 exists on all the back-end servers and importantly the partition index (**`ibm-slapdProxyPartitionIndex: 1`**) is the same on all back-end servers.

Example 13-23 Proxy configuration's `cn=ibmpolicies` partition definitions

```

dn: cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: cn=ibmpolicies split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 1
ibm-slapdProxyPartitionBase: cn=ibmpolicies
ibm-slapdProxySplitName: ibmpolicies
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
objectclass: top

dn: cn=split1, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split1
ibm-slapdProxyBackendServerDN: cn=Server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top

```

```

objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

dn: cn=split2, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split2
ibm-slapdProxyBackendServerDN: cn=Server2,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

dn: cn=split3, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split3
ibm-slapdProxyBackendServerDN: cn=Server3,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

dn: cn=split4, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split4
ibm-slapdProxyBackendServerDN: cn=Server4,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

```

The proxy configuration's `dc=com` partition definitions are shown in Example 13-24. From the first entry, the partition base is configured on `dc=com` with two partitions. The remaining two entries (second and third) define that the partitions of `dc=com` is split into on the first and second back-end servers by using an RDN hash split.

Example 13-24 Proxy configuration's `dc=com` partition definitions

```

dn: cn=dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: dc=com split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 2
ibm-slapdProxyPartitionBase: dc=com
ibm-slapdProxySplitName: dccomsplit

```

```

objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
objectclass: top

dn: cn=split1, cn=dc=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: split1
ibm-slapdProxyBackendServerDN: cn=Server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

dn: cn=split2, cn=dc=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: split2
ibm-slapdProxyBackendServerDN: cn=Server2,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 2
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

```

The proxy configuration's ou=dept1,dc=com partition definitions are shown in Example 13-25. From the first entry, the partition base is configured on ou=dept1,dc=com with a single partition. The remaining entry defines that the partition of ou=dept1,dc=com is put into the third back-end server by using a subtree split.

Example 13-25 Proxy configuration's ou=dept1,dc=com partition definitions

```

dn: cn=ou=dept1\,dc=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: ou=dept1,dc=com split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 2
ibm-slapdProxyPartitionBase: ou=dept1,dc=com
ibm-slapdProxySplitName: dept1split
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
objectclass: top

dn: cn=split1, cn=ou=dept1\,dc=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory,cn=Schemas,cn=Configuration
cn: split1
ibm-slapdProxyBackendServerDN: cn=Server3,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1

```

```
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit
```

The proxy configuration's ou=dept2,dc=com partition definitions are shown in Example 13-26. From the first entry, the partition base is configured on ou=dept2,dc=com with a single partition. The second entry defines that the partition of ou=dept2,dc=com is available on the third back-end server as a complete subtree.

Example 13-26 Proxy configuration's ou=dept2,dc=com partition definitions

```
dn: cn=ou\=dept2\,dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: ou=dept2,dc=com split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 2
ibm-slapdProxyPartitionBase: ou=dept2,dc=com
ibm-slapdProxySplitName: dept2split
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
objectclass: top

dn: cn=split1, cn=ou\=dept2\,dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split1
ibm-slapdProxyBackendServerDN: cn=Server4,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit
```

13.3 Fail over and load balancing

In a distributed directory environment, rather than having a single back-end server for each split, it is better to configure redundant back-end servers that contain the same data splits. When replication is configured between the back-end LDAP servers, it maintains the online updates to be replicated among those back-end servers. The proxy server configuration must be updated to create appropriate server groups and also to assign the servers with specific roles, such as primarywrite and so on.

In such an environment, the proxy server balances the load of read operations among the available back-end servers. The proxy server also handles the failover for write operations among the available master servers. If the primary write master server is not available for some reason, the proxy server automatically fails over to one of the remaining master servers. If the requested operation cannot be performed because the back-end server is not available, the proxy server returns an error on that operation. All subsequent requests fail over to the next available server.

Take a closer look at some of the key concepts:

► **High consistency**

In a high-consistency environment, the LDAP client applications demand that data is highly consistent. The application can update the data and within the same time frame read the data. In such environments, load balancing of read operations causes undesirable effects on the LDAP client applications. Instead, the proxy must use only the failover capabilities among the available servers. The proxy should direct both read and write operations to only one server at any particular time. If for some reason that server becomes unavailable, the proxy fails over to next available server for both read and write operations. In environments that require high consistency, it is sufficient to configure replication with all master server, and there is no need to configure read-only replica servers.

The proxy server handles load balancing on read requests when the high consistency feature is disabled. By default this feature is disabled. When high consistency is enabled, all read and write requests are sent to the primary write server until a failover occurs.

Enabling high consistency: Set the value of the **ibm-slapdProxyHighConsistency** configuration attribute to true.

► **Auto Failback feature**

When the Auto Failback feature is enabled, it makes the proxy server start by using the back-end server as soon as it becomes available.

If the Auto Failback feature is disabled, the back-end servers must be restored by using the Resume Role extended operation. The following are exceptions, when the proxy automatically restores even when the auto failback is disabled:

- If all back-end servers in a partition go down, the proxy server automatically restores the first available read server if it is the first one that comes online. Then, upon the availability of the first write server within the partition, the proxy automatically restores that, too. But if a write server comes online first, that is the only server that is automatically restored by the proxy, because the write master can handle both read and write operations.
- If all writable back-end servers in a partition go down, the proxy automatically restores the first write master that becomes available.

Note: Auto Failback can be enabled or disabled by setting the value of the **ibm-slapdEnableAutoFailBack** attribute to true or false.

► **Auto Failback based on replication queue size**

In a distributed directory environment, a back-end server might be taken offline for maintenance purposes. The proxy server fails over to other available servers. When the back-end server becomes available, the replication queues might contain many changes to be replicated to that server. This also means that the data on the server differs from that on other servers. In such cases, if that server is automatically restored by proxy, that might cause problems in the LDAP client applications. Rather than immediately automatically restoring the server, the proxy should wait until the changes are replicated to that server.

The proxy server provides a feature to enable failback, based on a configurable replication queue size. This feature enables automatic failback only when the replication queue size from the current write server to the server that fails back is less than or equal to the configured replication queue size.

Note: The `ibm-slapdProxyFailbackBasedOnQueueEnabled` configuration attribute enables auto failback, based on replication queue size.

The `ibm-slapdProxyFailbackQueueThreshold` configuration attribute sets the threshold value of replication queue size.

► **Health check feature**

The proxy server use health check feature to identify the available and down servers. The proxy runs a separate thread that determines the health check. Root Directory Server Entry (DSE) searches are initiated for the `ibmslapdisconfigurationmode` attribute against each of the back-end servers. If the search against any server fails, either because the server is down or it is in configuration-only mode, the failover process starts and marks the server as unavailable.

The health check feature can also detect when back-end servers become unresponsive. The proxy server can keep a count of outstanding search requests made to specific back-end server and find that its not getting any response. To enable this, set the `ibm-slapdProxyHealthCheckOlimit` attribute in a back-end server entry of the proxy configuration. The value of this attribute indicates the threshold for the number of outstanding health check requests before the proxy server determines that a back-end server is unresponsive.

Caution: `ibm-slapdProxyHealthCheckOlimit` specifies the `olimit` on health check. If the proxy server is under heavy load and the `olimit` value set is too small, the proxy might falsely report that the back-end server is unresponsive. To correct this problem, increase the `olimit`. However, the `olimit` value must be at least 3 less than the value of the connection pool size.

► **Health check status interval**

By default, the health check feature is disabled. The `ibm-slapdStatusInterval` attribute in back-end server entry of proxy configuration, represents the time interval (in seconds) between health check runs scheduled by the proxy server. The default value of this attribute is set to 0. The value 0 disables the health check. Its administrator's responsibility to find a suitable health check status interval for the environment, based on the back-end server's performance and load.

► **Weighted prioritization of back-end servers**

In a distributed directory environment, the back-end servers can be prioritized into five possible tiers. At a specified time, the proxy server uses only servers in one tier. When all of the write servers within a tier fail, the proxy server fails over to the second tier. When the second tier fails, it fails over to the third tier, and so on.

Weighted prioritization is configurable for each back-end server within a split. This is done by setting a value for the `ibm-slapdProxyTier` attribute. The default value for this attribute is 1. If the attribute is not present, the proxy treats the back-end server as a Tier 1 server. Valid values for this attribute range from 1 through 5.

During startup, all servers in all tiers are contacted. If the administrator wants the proxy server to start even if some of the back-end servers in different tiers are not available, server groups can be used.

13.3.1 Replicated back-ends

Consider a scenario in which a distributed directory environment is set up with a proxy server, along with three back-end servers, in a single back-end configuration. All of the three back-end servers are master servers that replicate the same data set. The replication is set up in both `dc=com` and `cn=ibmpolicies` contexts. The proxy server balances the load and failover functions in such an environment. This scenario with replicated back-ends is represented in Figure 13-5.

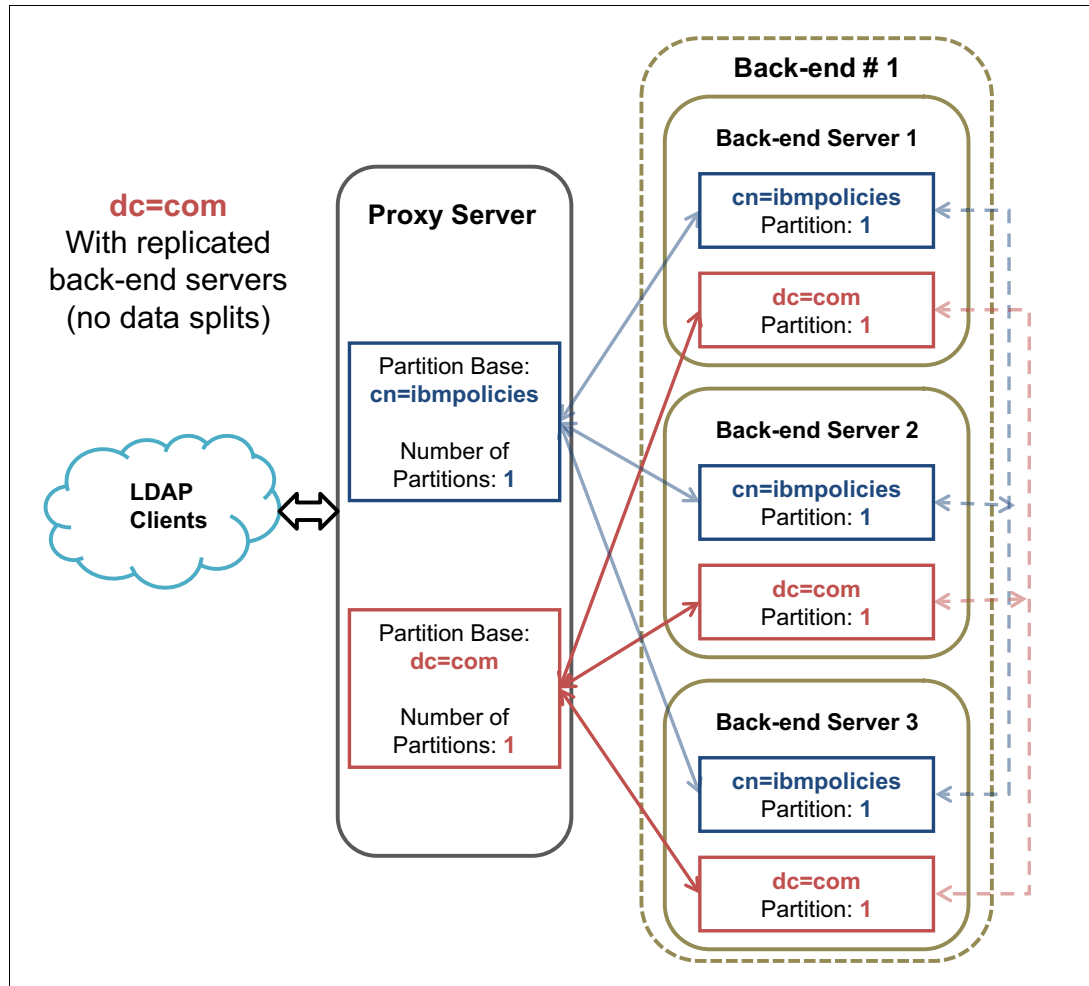


Figure 13-5 Replicated back-end scenario

LDAP requests from clients coming into the proxy server on `dc=com` and `cn=ibmpolicies` suffixes are being routed to one of the back-end servers. All update operations are routed to one of the back-end servers. The receiving server takes care of sending the same update to the remaining back-end servers through replication to keep the back-end servers in sync within the back end.

Configuration details

Examine the configuration of this scenario closely in terms of snippets of LDIF from proxy LDAP server instance's `ibmslapd.conf` file.

Proxy configuration's back-end entry is seen in Example 13-27 on page 122. The suffixes `dc=com` and `cn=ibmpolicies` are configured in this entry through `ibm-slapdSuffix`. Because

all of the back-end servers contain the same data set, distributed groups are not present in this scenario.

Example 13-27 Proxy configuration's back-end entry

```
dn: cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas, cn=Configuration
cn: Proxy Backend
ibm-slapdDNPartitionPlugin: libldapdnhash.<extn> dnHashInit
ibm-slapdPagedResAllowNonAdmin: TRUE
ibm-slapdPagedResLmt: 3
ibm-slapdPlugin: database libback-proxy.<extn> proxy_backend_init
ibm-slapdPlugin: extendedop libback-proxy.<extn> initResumeRole
ibm-slapdProxyEnableDistDynamicGroups: false
ibm-slapdProxyEnableDistGroups: false
ibm-slapdSuffix: dc=com
ibm-slapdSuffix: cn=ibmpolicies
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackend
```

The proxy configuration's back-end server entries are shown in the Example 13-28.

Example 13-28 Proxy configuration's back-end server entries

```
dn: cn=Server1, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server1
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckOlimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backendserver1:389
ibm-slapdServerID: 395e8a82-57f8-448c-bb59-953aa8ad6143
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry

dn: cn=Server2, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server2
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckOlimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backendserver2:389
ibm-slapdServerID: 014ac3eb-1aa2-4682-9cd3-4dbd2548bd44
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry
```

```
dn: cn=Server3, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: Server3
ibm-slapdProxyBindMethod: Simple
ibm-slapdProxyConnectionPoolSize: 5
ibm-slapdProxyHealthCheckLimit: 24
ibm-slapdProxyMaxPendingOpsPerClient: 5
ibm-slapdProxyDN: cn=root
ibm-slapdProxyPW: {AES256}6czMa3k6Tf3DJxyQo55cVw==
ibm-slapdProxyTargetURL: ldap://backendserver3:389
ibm-slapdServerID: 9f070e27-7351-4380-955c-2aa5431bf3b9
ibm-slapdStatusInterval: 0
objectClass: top
objectClass: ibm-slapdProxyBackendServer
objectClass: ibm-slapdConfigEntry
```

The proxy configuration's serverGroup entries definition is seen in Example 13-29.

Example 13-29 Proxy configuration's server group entries definition

```
dn: cn=serverGroup1, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: serverGroup1
ibm-slapdProxyBackendServerDN: cn=Server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerDN: cn=Server2,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerDN: cn=Server3,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendServerGroup
```

The proxy configuration's cn=ibmpolicies partition definitions are shown in Example 13-30. The partition base (**ibm-slapdProxyPartitionBase**) is configured on cn=ibmpolicies with a single partition (**ibm-slapdProxyNumPartitions: 1**). The remaining three entries define that the partition 1 exists on all three back-end servers and importantly the partition index (**ibm-slapdProxyPartitionIndex: 1**) is the same on all back-end servers.

Example 13-30 Proxy configuration's cn=ibmpolicies partition definitions

```
dn: cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory,
cn=Schemas, cn=Configuration
cn: cn=ibmpolicies split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 1
ibm-slapdProxyPartitionBase: cn=ibmpolicies
ibm-slapdProxySplitName: ibmpolicies
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
objectclass: top
```

```
dn: cn=split1, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split1
ibm-slapdProxyBackendServerDN: cn=Server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit
```

```
dn: cn=split2, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split2
ibm-slapdProxyBackendServerDN: cn=Server2,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit
```

```
dn: cn=split3, cn=cn\=ibmpolicies split, cn=ProxyDB, cn=Proxy Backends, cn=IBM
Directory, cn=Schemas, cn=Configuration
cn: split3
ibm-slapdProxyBackendServerDN: cn=Server3,cn=ProxyDB,cn=Proxy Backends,cn=IBM
Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit
```

The proxy configuration's `dc=com` partition definitions are shown in Example 13-31. The partition base (**`ibm-slapdProxyPartitionBase`**) is configured on `dc=com` with single partition (**`ibm-slapdProxyNumPartitions: 1`**). The remaining three entries define that the same single partition exists on all the three back-end servers with the partition index (**`ibm-slapdProxyPartitionIndex`**) value of 1.

Example 13-31 Proxy configuration's `dc=com` partition definitions

```
dn: cn=dc\=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas,
cn=Configuration
cn: dc=com split
ibm-slapdProxyAutoFailBack: false
ibm-slapdProxyFailbackBasedOnQueueEnabled: false
ibm-slapdProxyFailbackQueueThreshold: 5
ibm-slapdProxyHighConsistency: false
ibm-slapdProxyNumPartitions: 1
ibm-slapdProxyPartitionBase: dc=com
ibm-slapdProxySplitName: dccom
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplitContainer
```

objectclass: top

dn: cn=split1, cn=dc=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas, cn=Configuration
cn: split1
ibm-slapdProxyBackendServerDN: cn=Server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

dn: cn=split2, cn=dc=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas, cn=Configuration
cn: split2
ibm-slapdProxyBackendServerDN: cn=Server2,cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

dn: cn=split3, cn=dc=com split, cn=ProxyDB, cn=Proxy Backends, cn=IBM Directory, cn=Schemas, cn=Configuration
cn: split3
ibm-slapdProxyBackendServerDN: cn=Server3,cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,cn=Configuration
ibm-slapdProxyBackendServerRole: any
ibm-slapdProxyPartitionIndex: 1
ibm-slapdProxyTier: 1
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-slapdProxyBackendSplit

Note: In a similar way, the replicated back ends can be configured even when data is split among multiple back ends.

13.4 Tuning the proxy server

In this section, we focus on the proxy server configuration-tuning aspects. There are many tunable parameters in different sections of the proxy server configuration (`ibmslapd.conf`). The following subsections describe the proxy server-specific tunable parameters in detail per each configuration entry. Regular tunable parameters that are not specific to proxy server are not discussed in this section.

13.4.1 Front-end environment variables

In a proxy server configuration's (`ibmslapd.conf`) front-end entry, the environment variables that are related to the proxy server can be defined. Alternatively, the same environment

variables can be set in the shell where from the **ibmslapd** process starts. It is best to have these configured within the conf file. None of these front-end environment variables are dynamic. Therefore, any change to these variables is effective only when the proxy **ibmslapd** process restarts.

ODBCCONN

In a proxy server configuration, **ODBCCONN**, can be set in the front end to change the number of threads in proxy server process. If the proxy server is required to scale to handle multiple client requests simultaneously, it is desirable to change the value of **ODBCCONN**. If this environment variable is not set, the default value of 15 threads is initiated during the startup of **ibmslapd**. It is best to leave the default value for most proxy server environments (which means that there is no need to set this variable). The following LDIF snippet can be used to set **ODBCCONN** in the front-end entry through an LDAP modify operation:

```
dn: cn=Front End, cn=Configuration
changetype: modify
add: ibm-slapdSetenv
```

```
ibm-slapdSetenv: ODBCCONN=30
```

When you are increasing **ODBCCONN**, be sure to increase the connection pool size proportionally.

PROXY_CACHE_GAG_PW

The proxy server can cache the global administrator user's passwords locally within the proxy **ibmslapd** process. If password policy is enabled, caching of the Global Admin Group member passwords is disabled. If password policy is disabled, the caching of Global Admin Group members is enabled. **PROXY_CACHE_GAG_PW** environment variable can override this default behavior. **PROXY_CACHE_GAG_PW** set to YES enables password caching. **PROXY_CACHE_GAG_PW** set to any other value disables password caching. When the environment variable is not set, the default behavior is governed by the password policy setting. The following LDIF snippet can be used to set **ODBCCONN** in the front-end entry through LDAP modify operation:

```
dn: cn=Front End, cn=Configuration
changetype: modify
add: ibm-slapdSetenv
```

```
ibm-slapdSetenv: PROXY_CACHE_GAG_PW=YES
```

PROXY_GLOBAL_GROUP_PERIOD

By default, the proxy server updates the members of Global Admin Group by searching against the back-end servers every 30 sec. By setting the **PROXY_GLOBAL_GROUP_PERIOD** environment variable to a value (in seconds), the proxy search interval can be altered. The default value for this variable is 30 seconds. The default value takes effect when the variable is not set. In an environment where the Global Admin Group changes are rare, it is sufficient to search for Global Admin Group members once a day. In an environment where Global Admin Group changes are dynamic and can happen randomly, we suggest that you do not to set this variable explicitly; use the default behavior. The following LDIF snippet can be used to set **PROXY_GLOBAL_GROUP_PERIOD** in the front-end entry through an LDAP modify operation:

```
dn: cn=Front End, cn=Configuration
changetype: modify
add: ibm-slapdSetenv
```

```
ibm-slapdSetenv: PROXY_GLOBAL_GROUP_PERIOD=30
```

PROXY_USE_SINGLE_SENDER

During the startup, the proxy **ibm-slapd** process creates a sender thread per back end. These threads are used to send requests from proxy server over to back-end servers. Then, the **PROXY_USE_SINGLE_SENDER** environment variable changes this behavior and makes the proxy to use single sender thread to send requests to all back-end servers. When this variable is not defined, the default value of **false** becomes effective. This also means there will be individual sender threads for each back-end server. It is best not to set this variable and to use the default of one thread per back-end server. The following LDIF snippet can be used to set **PROXY_USE_SINGLE_SENDER** in the front-end entry through an LDAP modify operation:

```
dn: cn=Front End, cn=Configuration
changetype: modify
add: ibm-slapdSetenv
```

```
ibm-slapdSetenv: PROXY_USE_SINGLE_SENDER=false
```

PROXY_RECONNECT_TIME

The proxy server tries to reconnect to a back-end server, which is down. The default reconnect interval is 5 sec. To change this behavior, set the **PROXY_RECONNECT_TIME** environment variable to an appropriate value in seconds. If this variable is not set then the default value takes effective. It is best not to set this value and to take the default reconnect time of 5 sec. The following LDIF snippet can be used to set **PROXY_RECONNECT_TIME** in the front-end entry through an LDAP modify operation:

```
dn: cn=Front End, cn=Configuration
changetype: modify
add: ibm-slapdSetenv
```

```
ibm-slapdSetenv: PROXY_RECONNECT_TIME=5
```

LDAP_LIB_WRITE_TIMEOUT

The **LDAP_LIB_WRITE_TIMEOUT** environment variable specifies the time (in seconds) to wait for a socket to complete the current write operation and be write-ready. If this environment variable is not set, the default value of 300 sec (5 min) takes effect. In most environments, this default value is sufficient, so it is better not to set this environment variable. The following LDIF snippet can be used to set **LDAP_LIB_WRITE_TIMEOUT** in the front-end entry through an LDAP modify operation:

```
dn: cn=Front End, cn=Configuration
changetype: modify
add: ibm-slapdSetenv
```

```
ibm-slapdSetenv: LDAP_LIB_WRITE_TIMEOUT=300
```

FLOW_CONTROL_SLEEP_TIME

The proxy server can handle flow control among the connections that are established to the back-end servers. When there are no free back-end connections available, the proxy server temporarily suspends reading from socket. It then checks periodically to see whether there is a free back-end connection that has become available. The frequency with which this check is done is determined by the **FLOW_CONTROL_SLEEP_TIME** environment variable. This variable takes integer value in milliseconds and specifies the frequency with which the proxy checks the back-end connections. If the environment variable is not set, it defaults to 5. The following LDIF snippet can be used to set **FLOW_CONTROL_SLEEP_TIME** in the front-end entry through an LDAP modify operation:

```
dn: cn=Front End, cn=Configuration
changetype: modify
add: ibm-slapdSetenv

ibm-slapdSetenv: FLOW_CONTROL_SLEEP_TIME=30
```

13.4.2 Tuning attributes in Proxy Backend database entry:

This section describes various tunable attributes from the Proxy Backend database entry of the proxy's configuration.

Tip: *Dynamically changed attributes* are attributes that can be changed dynamically. There is no need to restart the server for these changes to take effect. If you use the command-line **LDAPmodify** to update these attribute values, after modification, the **ldapexop -op readconfig** option must be sent to the LDAP server process to re-read the configuration, and the attribute change is effective immediately.

ibm-slapdPagedResAllowNonAdmin

A paged search operation against an LDAP server uses a server thread until the paged search operation is finished. When enabling this attribute, be careful in tuning the **ibm-slapdPagedResLmt** attribute also. The **ibm-slapdPagedResAllowNonAdmin** attribute is an optional attribute that can be in Proxy Backend database entry of a proxy configuration (`ibmslapd.conf`). This attribute is a dynamically changeable attribute. When the value of this attribute is set to **TRUE**, the proxy allows non-administrative users (including anonymous binds) to perform paged searches against the proxy server. When this attribute value is set to **FALSE**, only administrative users can do the paged searches against the proxy server. Non-administrative binds are rejected with an Insufficient Access Rights notice. If this attribute is not set in the configuration, then the attribute value of **FALSE** takes effect by default.

The following LDIF snippet can be used to set through an LDAP modify operation:

```
dn: cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,cn=Configuration
changetype: modify
replace: ibm-slapdPagedResAllowNonAdmin

ibm-slapdPagedResAllowNonAdmin: TRUE
```

ibm-slapdPagedResLmt

The optional **ibm-slapdPagedResLmt** attribute can be in Proxy Backend database entry. This attribute defines maximum number of allowed simultaneous active (or outstanding) paged search requests. By default, the server allows a maximum of three outstanding simple paged search operations at any time. To ensure the fastest response for subsequent simple paged result requests, the server holds a database connection open throughout the duration of the search request until the user cancels the simple paged results request or until the last result is returned to the client application. This administrative limit is designed to avoid denied service due to flooding of paged search requests.

It is best to set the **ibm-slapdPagedResLmt** value lower than the maximum number of proxy server threads (**ODBCCONN**). If this attribute does not exist, add or set the required value. If this attribute does not exist, the default value of 3 takes effect during server startup. This attribute is a dynamically changeable attribute. To change the value of this administrative limit, use the following LDIF snippet through an LDAP modify operation:


```
dn: cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,cn=Configuration
changetype: modify
replace: ibm-slapdPagedResLmt
```

```
ibm-slapdPagedResLmt: 3
```

ibm-slapdProxyEnableDistGroups

A distributed group is a group in which the member or uniquemember attributes may contain user DN values that do not exist on the same back-end server for the group.

ibm-slapdProxyEnableDistGroups is a required Boolean attribute in the Proxy Backend database entry, which can take either TRUE or FALSE value. If the value is set to TRUE, the evaluation of all groups by the proxy server in distributed directory environment takes place.

This means that the proxy gets user group evaluations from individual back-end servers and provides effective output to the client application. Enable this attribute only if the distributed groups are present in the back-end distributed directory environment. If the distributed groups are not present it is suggested to disable this feature by setting a value of FALSE for **ibm-slapdProxyEnableDistGroups**. Typically this attribute is set to true or false during the initial configuration of proxy server. Change in back-end data distribution regarding groups should result in reevaluation of this attribute value. Also, changing this attribute requires that the proxy server be restarted to get the latest value. To change the value of this attribute, use the following LDIF snippet through an LDAP modify operation:

```
dn: cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,cn=Configuration
changetype: modify
replace: ibm-slapdProxyEnableDistGroups
```

```
ibm-slapdProxyEnableDistGroups: TRUE
```

ibm-slapdProxyEnableDistDynamicGroups

A distributed dynamic group is a dynamic group in which the member URL might evaluate user DN values that do not exist on the same back-end server for the group.

ibm-slapdProxyEnableDistDynamicGroups is a required Boolean attribute in the Proxy Backend database. It can take either a TRUE or FALSE value. If the value is set to TRUE, the evaluation of all dynamic groups by the proxy server in distributed directory environment takes place. This means that the proxy gets user group evaluations from individual back-end servers and provides effective output to the client application. Enable this attribute only if the distributed dynamic groups are present in the back-end distributed directory environment. If the distributed dynamic groups are not present, disable this feature by setting the **ibm-slapdProxyEnableDistDynamicGroups** value to FALSE.

Typically, this attribute is set to true or false during the initial configuration of proxy server. Change in back-end data distribution regarding groups should result in reevaluation of this attribute value. Changing this attribute also requires the proxy server to be restarted to get the latest value. To change the value of this attribute, use the following LDIF snippet through an LDAP modify operation:

```
dn: cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,cn=Configuration
changetype: modify
replace: ibm-slapdProxyEnableDistDynamicGroups
```

```
ibm-slapdProxyEnableDistDynamicGroups: TRUE
```

13.4.3 Tuning attributes in Backend server entry

This section describes various tunable attributes from back-end server entry of proxy server's configuration.

ibm-slapdProxyConnectionPoolSize

In proxy configuration's back-end server entry, the **ibm-slapdProxyConnectionPoolSize** attribute determines the number of connections that the proxy server can have with the back-end server. The minimum value is 1 and the maximum value is 100. The default value is 5. Do not set the value in the Connection pool size field to less than 5. The number of connections to the back-end server should be less than or equal to the number of workers that are configured on the back-end server. Any changes to this attribute require the proxy server to be restarted to put the new value into effect. To change the value of this attribute, use the following LDIF snippet through an LDAP modify operation:

```
dn: cn=server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,  
cn=Configuration  
changetype: modify  
replace: ibm-slapdProxyConnectionPoolSize
```

```
ibm-slapdProxyConnectionPoolSize: 15
```

ibm-slapdStatusInterval

The **ibm-slapdStatusInterval** attribute represents the time interval (in seconds) between health check runs scheduled by the server. This attribute is not a dynamic attribute and when its not set explicitly, the default value of 0 takes effect, which also disables the health check. To change the value of this attribute, use the following LDIF snippet through an LDAP modify operation:

```
dn: cn=server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,  
cn=Configuration  
changetype: modify  
replace: ibm-slapdStatusInterval
```

```
ibm-slapdStatusInterval: 15
```

ibm-slapdProxyMaxPendingOpsPerClient

The **ibm-slapdProxyMaxPendingOpsPerClient** attribute is used to configure the threshold limit for pending requests from a client connection in a back-end connection. On reaching this threshold limit, requests from the client connection are not read until the pending requests in the back-end connection reduce to a value below the specified threshold limit. If this attribute is not specified, the maximum pending client operations defaults to 5. If a value of 0 is assigned to the **ibm-slapdProxyMaxPendingOpsPerClient** attribute, the number of client operations per connection that is pending can be unlimited. This attribute is not a dynamically changeable attribute. To change the value of this attribute, use the following LDIF snippet through an LDAP modify operation:

```
dn: cn=server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,  
cn=Configuration  
changetype: modify  
replace: ibm-slapdProxyMaxPendingOpsPerClient
```

```
ibm-slapdProxyMaxPendingOpsPerClient: 5
```

ibm-slapdProxyHealthCheckOlimit

When the **ibm-slapdProxyHealthCheckOlimit** attribute is set, it enables the health check feature of the proxy server. The value of this attribute indicates the threshold for the number of outstanding health check requests before the proxy server determines that a back-end server is unresponsive. Set this attribute value to be at least 3 points less than the value of the connection pool size. This attribute is not a dynamically changeable attribute. To change the value of this attribute, use the following LDIF snippet through an LDAP modify operation:

```
dn: cn=server1,cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,  
cn=Configuration  
changetype: modify  
replace: ibm-slapdProxyHealthCheckOlimit
```

```
ibm-slapdProxyHealthCheckOlimit: 5
```

13.4.4 Tuning attributes in partition base entry

This section describes various tunable attributes from Partition base entry of proxy server's configuration.

ibm-slapdProxyAutoFailBack

Proxy server's Auto Failback feature is described previously in this chapter, under 13.3, "Fail over and load balancing" on page 118. The **ibm-slapdProxyAutoFailBack** attribute is used to enable or disable the proxy server's Auto Failback feature. If the value of this attribute is set to TRUE, Auto Failback is enabled and if its set to FALSE, Auto Failback is disabled. The default value of **ibm-slapdEnableAutoFailBack** is TRUE. Also, the default value takes effect when this attribute is not defined explicitly in the proxy server's partition base entry. This is not a dynamically changeable attribute. To change the value of this attribute, here is an example of an LDIF snippet that can be applied through an LDAP modify operation:

```
dn: cn=dc\com split,cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,  
cn=Configuration  
changetype: modify  
replace: ibm-slapdProxyAutoFailBack
```

```
ibm-slapdProxyAutoFailBack: TRUE
```

ibm-slapdProxyHighConsistency

Proxy server's high consistency feature is was described previously in this chapter, under 13.3, "Fail over and load balancing" on page 118. The **ibm-slapdProxyHighConsistency** attribute is used to enable or disable the proxy server's high consistency feature. If the value of this attribute is set to TRUE, high consistency is enabled and the proxy server sends all read and write requests to the primary write back-end server. If the attribute value is set to FALSE, high consistency is disabled, and the proxy server balances the load of read operations among all available back-end servers. The default value of **ibm-slapdProxyHighConsistency** is FALSE. Also, the default value takes effect when this attribute is not defined explicitly in the proxy server's partition base entry. This is not a dynamically changeable attribute.

To change the value of this attribute, here is an example of an LDIF snippet that can be applied through an LDAP modify operation:

```
dn: cn=dc\=com split,cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,
cn=Configuration
changetype: modify
replace: ibm-slapdProxyHighConsistency
```

```
ibm-slapdProxyHighConsistency: TRUE
```

ibm-slapdProxyFailbackBasedOnQueueEnabled

The attribute **ibm-slapdProxyFailbackBasedOnQueueEnabled** is used to enable or disable the proxy server feature of “fail-back based on a configurable replication queue size.” If the value of **ibm-slapdProxyFailbackBasedOnQueueEnabled** is set to TRUE, this feature is enabled. If it is set to FALSE, the feature is disabled. The default value is TRUE. Also, the default value takes effect when this attribute is not defined explicitly in the proxy server’s partition base entry.

This is not a dynamically changeable attribute. To change the value of this attribute, here is an example of an LDIF snippet that can be applied through an LDAP modify operation:

```
dn: cn=dc\=com split,cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,
cn=Configuration
changetype: modify
replace: ibm-slapdProxyFailbackBasedOnQueueEnabled
```

```
ibm-slapdProxyFailbackBasedOnQueueEnabled: TRUE
```

ibm-slapdProxyFailbackQueueThreshold

When the “fail-back based on configurable replication queue size” feature is enabled (**ibm-slapdProxyFailbackBasedOnQueueEnabled: TRUE**) in the proxy server’s partition base entry, the proxy server uses the **ibm-slapdProxyFailbackQueueThreshold** attribute value to set the replication queue threshold size. The default value of this attribute is set to 5. This means that the fail back happens only if the replication queue size is less than or equal to 5. Also, the default value takes effect when this attribute is not defined explicitly in the proxy server’s partition base entry.

This is not a dynamically changeable attribute. To change the value of this attribute, here is an example of an LDIF snippet that can be applied through an LDAP modify operation:

```
dn: cn=dc\=com split,cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,cn=Schemas,
cn=Configuration
changetype: modify
replace: ibm-slapdProxyFailbackQueueThreshold
```

```
ibm-slapdProxyFailbackQueueThreshold: TRUE
```

13.4.5 Tuning attributes in partition entry

This section describes various tunable attributes from the Partition entry of the proxy server’s configuration.

ibm-slapdProxyBackendServerRole

The **ibm-slapdProxyBackendServerRole** attribute represents the role of a back-end directory server in a particular partition. The attribute is defined in a Partition entry with the **ibm-slapdProxyBackendSplit** object class. There are two values that can be assigned to this attribute: **primarywrite** or **any**. This is a required attribute and must be given a value when you define the proxy server’s partition entry.

This is not a dynamically changeable attribute. To change the value of this attribute, here is an example of an LDIF snippet that can be applied through an LDAP modify operation:

```
dn: cn=split1,cn=dc\=com split,cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,
cn=Schemas,cn=Configuration
changetype: modify
replace: ibm-slapdProxyBackendServerRole
```

```
ibm-slapdProxyBackendServerRole: primarywrite
```

ibm-slapdProxyTier

Weighted prioritization is configurable for each back-end server within a split. This is done by setting a value for the **ibm-slapdProxyTier** attribute. The default value for this attribute is 1. If the attribute is not present, the proxy treats the back-end server as a Tier 1 server. Valid values for this attribute range from 1 through 5.

This is not a dynamically changeable attribute. To change the value of this attribute, here is an example of an LDIF snippet that can be applied through an LDAP modify operation:

```
dn: cn=split1,cn=dc\=com split,cn=ProxyDB,cn=Proxy Backends,cn=IBM Directory,
cn=Schemas,cn=Configuration
changetype: modify
replace: ibm-slapdProxyTier
```

```
ibm-slapdProxyTier: 1
```




LDAP replication information

Replication is a technique that is used by directory servers to improve performance, availability, and reliability. The replication process keeps the data in multiple directory servers synchronized. In this chapter, we provide several tips for replication and performance-related considerations.

For more information, see the Replication section of the Administration Guide in the IBM Directory Server Information Center:

<http://ibm.co/0nrOMX>

Replication provides three main benefits:

- ▶ **Redundancy of information.** Replicas back up the content of their supplier servers.
- ▶ **Faster searches.** Search requests can be spread among several different servers, instead of a single server. This improves the response time for the request completion.
- ▶ **Security and content filtering.** Replicas can contain subsets of data in a supplier server.

This chapter covers the following replication topics:

- ▶ Replication performance considerations
- ▶ The importance of cn=ibmpolicies replication
- ▶ Distinguishing between LDAP reads and writes
- ▶ Conflict resolution
- ▶ Monitoring and managing replication
- ▶ Synchronizing two-way cryptography for server instances

14.1 Replication performance considerations

When you are defining your replication topology for optimal performance, there are several important aspects to consider:

- ▶ Cryptographic synchronization
- ▶ Replication by using SSL or not
- ▶ Conflict resolution
- ▶ Multithreaded (asynchronous) or single-threaded replication

14.1.1 Cryptographic synchronization

If Advanced Encryption Standard (AES) encryption is used in the environment and passwords or other encrypted attributes are updated frequently, it is important to make sure that the servers are cryptographically synchronized in the replication topology. This allows the LDAP servers to replicate the AES-encrypted value across the topology without adding the administration tasks for decrypting and re-encrypting the value. Depending on how frequently this activity occurs in the environment, this can improve replication performance and reduce potential processor overhead.

14.1.2 Replication by using SSL or not

Replication can be configured to use Secure Sockets Layer (SSL) or non-SSL, depending on the requirements of the environment. If replication over SSL is required, you might need to tune the timeout of the SSL handshake. The default value of the SSL handshake is 1000 milliseconds (1 second). In environments where there is a potential for network latency issues in case of peak loads, it is important to set and tune the **SSL_TIMEOUT_MILLISEC** environment variable. This variable allows the time out of the SSL handshake to be extended to a configurable value. It is set in the `cn=Front End, cn=Configuration` stanza of the `ibmslapd.conf` file, as shown in Example 14-1. After the variable is set, it is important to monitor the `ibmslapd.log` for SSL-related failures that might indicate that the value needs to be tuned further.

Example 14-1 ibmslapd.conf where the SSL time-out value is set to 5000 millisecs

```
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: SSL_TIMEOUT_MILLISEC=5000
objectclass: top
objectclass: ibm-slapdConfigEntry
objectclass: ibm-SlapdFrontEnd
```

To configure the **SSL_TIMEOUT_MILLISEC** variable, follow these steps:

1. Create an LDAP Directory Interchange Format (LDIF) file, as shown in Example 14-2 on page 137.

Example 14-2 Create an ssl.ldif file

```
-----ssl.ldif-----
dn: cn=Front End, cn=Configuration
changetype: modify
add: ibm-slapdSetEnv
ibm-slapdSetenv: SSL_TIMEOUT_MILLISEC=5000
-----
```

2. Apply the LDIF file by using this command:

```
idsldapmodify -D cn=root -w ? -i ss.ldif
```

3. Because an **ibm-slapdSetEnv** parameter was added, a server restart is required.

14.1.3 Conflict resolution

Starting with the Version 6.2 release of the Directory Server, there are two features that pertain to conflict resolution:

- ▶ Increased time stamp granularity
- ▶ Configuration parameter to selectively enable or disable conflict resolution for groups

The increased time stamp granularity from millisecond to microsecond (for non-Microsoft Windows platforms) helps to significantly reduce the potential for false conflicts. Conflict resolution for add and modify operations in peer-to-peer replication is based on the time stamp. The entry update with the most recent modified TimeStamp on any server in a multi-master replication environment is the one that takes precedence. Replicated delete and rename requests are accepted in the order received without conflict resolution. When a replication conflict is detected, the replaced entry is archived for recovery purposes in the Lost and Found log.

When conflict resolution is enabled and changes are made to the membership of a particular group on two servers concurrently, conflict resolution is triggered repeatedly. That can adversely affect the server's performance if the groups are large.

Starting with Directory Server Version 6.2, group conflict resolution is disabled by default. You can selectively enable or disable replication conflict resolution for modifications to group entries by defining the value of the **ibm-slapdEnableConflictResolutionForGroups** attribute, which is under the cn=Replication, cn=configuration entry of the configuration file.

If the **ibm-slapdEnableConflictResolutionForGroups** attribute is set to FALSE, no conflict resolution is attempted, even if a conflict is detected for operations on group entries, such as adding, deleting, or renaming a **member** or **uniquemember** attribute type.

However, a single modify request can target multiple attributes. For a single modify request, if any attribute other than **member** or **uniquemember** is used, replication conflict resolution is performed, even though the **ibm-slapdEnableConflictResolutionForGroups** attribute is set to FALSE.

14.1.4 Multithreaded (asynchronous) or single-threaded replication

Multithreaded replication, also referred to as *non-blocking* or *asynchronous* replication, allows the supplier to send several updates to the consumer without waiting for a response. Single-threaded replication updates are sent one at a time and are dependent upon a successful response to continue replicating. If an error is encountered, replication is blocked until the error is corrected or skipped, as appropriate. Multithreaded replication has a demanding set of requirements, so it requires careful monitoring and remediation of failures. If

the environment does not meet these stringent requirements, use single-threaded replication. Single-threaded replication does not require continuous monitoring or administration unless there are serious data mismatches or configuration errors.

The multithreaded replication function replaces the current single replication thread with a minimum of three threads to service the replication agreement:

- ▶ Main thread
- ▶ Sender thread
- ▶ Receiver thread

You can have 1 through 32 consumer connections. Set the number of consumer connections to match the number of processors on your machine. Using multiple threads enables the supplier to send the updates to the consumer without waiting on the response from the consumer.

Anyone with a replication backlog might consider switching to multithreaded replication in these candidate environments:

- ▶ A high update rate
- ▶ No down-level servers
- ▶ Common AES salt and sync if encryption is AES and passwords are updated often
- ▶ Small fan-out (for example, do not try 8 connections per agreement with 24 replicas)
- ▶ Available servers and reliable network
- ▶ Data consistency is not critical
- ▶ All replication schedules are immediate
- ▶ Multiprocessor machines

Multithreaded replication is difficult to administer if servers or networks are not reliable.

When errors occur, the errors are logged and can be replayed by the administrator, but the error logs must be monitored closely. Example 14-3 depicts a search to show the replication backlog for all agreements that are supplied by one server.

Example 14-3 Search to show the replication backlog for all agreements supplied by one server

```
idsldapsearch -h supplier-host -D cn=admin -w ? -s sub
  objectclass=ibm-replicationagreement
  ibm-replicationpendingchangeount ibm-replicationstate
```

14.2 The importance of cn=ibmpolicies replication

It is important to set up replication for the cn=ibmpolicies subtree. This allows for schema modifications to remain in sync with the rest of the replication topology. If this subtree is not configured for replication, there is the potential for data to become out of sync and for replication errors to occur on other data replication contexts.

For example, this can happen if a new attribute or object class is defined on a server in the replication topology and a new entry is added that uses the definition. If the attribute or objectclass definition does not exist on a consumer in the replication topology, adding the new user causes the replication queues to block the supplier.

14.3 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.

- ▶ You can point all read-only applications to a load-balancing virtual IP address and all read-write applications to a virtual IP address that is configured for failover, not load balancing.
- ▶ You can divide the 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 fails over to the other.
- ▶ You can use the Directory Server directory proxy, which can distinguish between LDAP reads and writes and can balance read loads while handling failover for writes.

With Directory Server Version 6 and later, replication can be multithreaded for any suffix except `cn=ibmpolicies`, which can be only single-threaded. This is not an issue, because `cn=ibmpolicies` suffix is used only for schema and password policy updates and does not have a use rate that is high enough to need multithreading. Multithreading 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 already looked for network bottlenecks, you might need to turn on multithreading for that suffix. Remember that each server in the replication agreement for that suffix must be configured to use multithreading to get the best results from turning on multithreading.

Replication helps with scaling for read traffic, but it does not help to scale writes. Every replica contains all entries, so every write goes to each replica. A single instance can support several hundred writes a second. Your situation can vary, depending on the size of the write, the number of indexes, the amount of memory, the speed of the machine, the speed and type of access to the storage disk, and the number of entries in your database.

The key is this: If you are going to have high write rates along with 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 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 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 add another partition later without dumping all of the data and reloading it all across all of the partitions again. This is because decisions about what data goes on which partition are determined by several factors, such as the number of partitions that are to be used and the hash process that is being used. Considerable planning is necessary to design an enclave that uses partitions.

A few applications, such as the IBM Security 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 server that is different from the one where the write occurred, and the update might not have been replicated to that server yet. 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.

14.4 Conflict resolution

Replication errors are rare occurrences in a correctly configured replication topology, but it is not safe to assume that they never occur. Directory Server Version 6.2 and later use fine-grained time stamps to track all updates made to the environment. The conflict resolution process uses these time stamps to resolve which change takes effect, based on the time stamp of the entry. This process has overhead associated with it and can trigger 10 retries before a conflict is discarded. If conflicts occur in group processing, the associated overhead can increase significantly. To avoid unnecessary overhead, it is important to understand when to use conflict resolution and when it is safe to disable it.

When using IBM Security Access Manager or any other application that directs all write operations to only one master at once or an application that is using a failover type load balancer between the application and Directory Server v6, you can turn this process off and save a substantial amount 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' time stamps for replicated entries in an attempt to resolve conflicts between the entries. Instead, it runs each request, one at a time, and changes the order in which they were received.

With *conflict resolution* turned on, if your Directory Server v6 servers report high CPU load and many entries in the `lostandfound.log` on each of the peer or forwarder servers, review the IBM Technote titled "Disabling replication conflict resolution:"

http://www.ibm.com/support/docview.wss?rs=767&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. Therefore, changing it while the server is running has no effect on the server.

We suggest that you turn off conflict resolution in your LDAP environment under these conditions:

- ▶ When you have more than two peers or forwarder replicas in your network
And
- ▶ You are using IBM Security Access Manager or any other product that writes to only one LDAP server and fails 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 must be done on each LDAP server in the enclave. You must restart the **ibmslapd** process for this to take effect. See Example 14-4 on page 141 for details.

Example 14-4 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
```

14.5 Monitoring and managing replication

Directory Server provides operational attributes and extended operations to aid in monitoring, managing, and debugging replication. Directory Server Version 6.1 and later versions include a feature to implement RFC 3673. For replication, two special attributes, **+ibmrep1** and **++ibmrep1i**, are defined to request replication-related operational attributes in a search. The **+** and **++** are subsets of the operational attributes. The single **+** is less expensive. In Table 14-1, the **++** includes all operational attributes that are shown in the **+ Attribute list** column and those listed in the **++** column.

As in earlier releases, you can still search and specify only the operational attributes that you want to see. The **+ibmrep1** and **++ibmrep1i** attributes are defined to reduce the complexity when you are searching for all of the operational attributes.

Table 14-1 Replication-related operational attributes in a search

+ Attribute list	++ Attribute list
+ibmrep1 returns all of the these	++ibmrep1 returns all of these
ibm-replicationChangeLDIF ibm-replicationLastActivationTime ibm-replicationLastChangeId ibm-replicationLastFinishTime ibm-replicationLastResult ibm-replicationLastResultAdditional ibm-replicationNextTime ibm-replicationPendingChangeCount ibm-replicationState ibm-replicationFailedChangeCount ibm-replicationperformance	ibm-replicationChangeLDIF ibm-replicationLastActivationTime ibm-replicationLastChangeId ibm-replicationLastFinishTime ibm-replicationLastResult ibm-replicationLastResultAdditional ibm-replicationNextTime ibm-replicationPendingChangeCount ibm-replicationState ibm-replicationFailedChangeCount ibm-replicationperformance ibm-replicationPendingChanges ibm-replicationFailedChanges

To search on a specific agreement, use the following command:

```
idsldapsearch -h hostName -p <port> -D <adminDN> -w <password> -b
<ReplicationAgreement> objectclass=* ++ibmrep1
```

For example:

```
idsldapsearch -h peer1 -p 1389 -D cn=root -w secret -b  
cn=peer2:2389,cn=peer1:1389,ibm-replicaGroup=default,0=IBM,C=US objectclass=*  
++ibmrepl
```

To search *all* agreements, use the following command:

```
idsldapsearch -h hostName -p <port> -D <adminDN> -w <password> -s sub -b " "  
objectclass=ibm-replicationagreement ++ibmrepl
```

For example:

```
idsldapsearch -h peer1 -p 1389 -D cn=root -w secret -s sub -b " "  
objectclass=ibm-replicationagreement ++ibmrepl
```

14.5.1 Operational attributes

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

► **ibm-replicationLastActivationTime**

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

► **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 that was sent to this consumer.

► **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 a 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.

► **ibm-replicationLastResult**

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

ibm-replicationPendingChangeCount

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

► **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.

► **ibm-replicationChangeLDIF**

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

Note: Single-threaded replication only.

► **ibm-replicationFailedChanges**

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

► **ibm-replicationFailedChangeCount**

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

► **ibm-replicationIsQuiesced**

This attribute returns a Boolean value that indicates whether the subtree has been quiesced. The base DN of the search should be **replicationContext**.

14.5.2 Environment variables for use with replication

The best method for specifying environment variables is to update the dn: cn=Front End,cn=Configuration stanza of the configuration file (**ibmslapd.conf**) by using the **ibm-slapdSetenv** configuration attribute to specify the environment variable and value by using the command-line utilities.

► **IBMSLAPD_REPL_CHUNK_SIZE**

Set the number of updates to retrieve by each call to the RDBM back end. The default number of updates that can be retrieved is 100. This default can be changed by using this variable.

► **IBMSLAPD_REPL_IDENTIFY_UPDATES**

If it is set to any value, the supplier server sends the **LDAP_REPL_UPDATE_ID** control. See **ReplicationEnhancements_accelerated HLD** for more information on the control.

► **IBMSLAPD_REPL_READY_SIZE_LIMIT**

This environment variable sets the maximum number of operations to hold in the replication queues. The default number of operation that can be held in a replication queue is 10. This default can be changed by using this variable.

► **IBMSLAPD_REPL_UPDATE_EXTRA_SECS**

This environment variable sets the number of seconds to be added to the time limit for updates to be replicated. The value cannot be less than zero. Without this variable, the time limit is 60 seconds.

► **IBMSLAPD_REPL_WAIT_ON_DEPENDENCY**

If this flag is not set, the consumer servers do not process operations that are received on the same connection in the order in which they were received. If this flag is set and there is a dependency on another operation still in the queue, the server waits for it to complete. The performance degrades if the variable is set.

► **SSL_TIMEOUT_MILLISEC**

The server waits for the value that is specified in this variable to complete SSL handshake. Sometimes, clients that are connecting to servers in a high latency network might fail during the SSL handshake. This variable can be used to change the handshake timeout from the default 1000 ms (1 sec). This might be required in high-latency environments where replication is configured over SSL.

14.5.3 Troubleshooting replication problems

Consider these 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.
- Review the replication status on the supplier for the agreement that is experiencing errors.

To search for a specific agreement, use the following command:

```
idsldapsearch -h hostName -p <port> -D <adminDN> -w <password> -b  
<ReplicationAgreement> objectclass=* ++ibmrep1
```

For example:

```
idsldapsearch -h peer1 -p 1389 -D cn=root -w <password> -b  
cn=peer2:2389,cn=peer1:1389,ibm-replicaGroup=default,0=IBM,C=US objectclass=*  
++ibmrep1
```

- Start with the supplier's log file.
 - Check to make sure that it can connect to the consumer.
 - Check whether it is able to bind correctly.
 - Check for replication-related error messages.

14.6 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 get 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 must perform this procedure (**idsgendirksf**) *before* you do any of the following tasks:

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

If you are creating a new instance, you can cryptographically synchronize it with the source instance by specifying the source instance's encryption seed and encryption salt value at instance creation time.



Adding a new LDAP server to an existing enclave

In this chapter, we explain how to install and add a new Lightweight Directory Access Protocol (LDAP) server to an existing replicated LDAP server environment. We cover the following topics:

- ▶ Installing a new Directory Server
- ▶ Adding a new LDAP server to replication topology
- ▶ Using the `idsideploy` method to add a new LDAP server
- ▶ Testing replication

15.1 Installing a new Directory Server

On the new system that is going to be used for the LDAP server, complete these tasks:

1. Install IBM Directory Server and its required components by using either the InstallShield GUI method or the operating system installation method.
2. Install the latest IBM DB2 and Directory Server Fix Packs before you configure the new server.
3. Build this new server the same way that the other LDAP servers were built, using the same user and group IDs, LDAP and DB2 instance names, instance location path, and database location path.
4. Create a new instance, configure the associated database, add the required suffixes to the configuration, and then configure the admin DN and password.

15.1.1 Update the server ID in the LDAP server configuration

After configuration of the LDAP server process on the first startup, an `ibm-slappedServerId` attribute that looks something like this gets added to the `ibmslapd.conf` file:

```
4dbdc73a-2476-4039-8808-9655f95d917
```

Optionally, you can change the server ID in the `ibmslapd.conf` file for easier identification and readability. If you have a complex enclave of multiple replicated systems, having a recognizable server ID helps in troubleshooting. It is better that you use a server ID that better describes the server you are working with. Check the server ID values from the current set of servers and prefer to use the same convention.

To update (add or modify) the server ID, start the server in configuration only mode and then use the `idsldapmodify` operation with the LDAP Directory Interchange Format (LDIF) that is shown in Example 15-1.

Important: The server ID plays an important role in the replication topology, because it identifies the server and its role. The server ID specified in `ibmslapd.conf` must match the server ID used in the replication agreement. The suggested format for the server ID is `<shorthostname>-uid`.

Example 15-1 Updating the server ID in `ibmslapd.conf` file by using the `ldapmodify` method

```
###Start the ibmslapd in configuration only mode.
==> ibmslapd -I <instanceName> -a
GLPSRV034I Server starting in configuration only mode.
...
GLPCOM003I Non-SSL port initialized to 389.

###Create a file confmod.ldif with the four lines that follow, using the correct
server ID.
==> cat confmod.ldif
dn: cn=Configuration
changetype: modify
replace: ibm-slappedServerId
ibm-slappedServerId: newldapshorthostname-uid

###Use the ldap modify operation with the confmod.ldif to update the server ID.
==> idsldapmodify -p 389 -D cn=root -w passwd -i confmod.ldif
```

Operation 0 modifying entry cn=Configuration

```
###Stop the ibmslapd.  
==> ibmslapd -I <instanceName> -k
```

15.2 Adding a new LDAP server to replication topology

To add the new LDAP server to the existing replication topology, complete the tasks in the following sections.

15.2.1 Define the role of the new LDAP server

Based on the *replication*, the new LDAP server can assume one of the following roles in the replication topology.

- ▶ Peer server (master server)
- ▶ Replica server
- ▶ Forwarder
- ▶ Gateway

Based on the *role* in the replication, the new LDAP server can assume either a *supplier* or *consumer* function. For example, a peer server is a supplier server for all other servers in the replication topology, but it is also a consumer for any remaining peer servers.

15.2.2 Define the default credential object and referral that is based on the role

Based on the role of the LDAP server, create an LDIF file with the contents that follow to add the default credential and referral to the new server's configuration.

1. The LDIF file for a peer server has the following contents:

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

2. For a replica server, the LDIF file contains these elements:

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

3. Use the **ldapadd** command with appropriate LDIF:

```
idsldapadd -p 389 -D cn=root -w passwd -i <ldif_file>
```

15.2.3 Create the replication settings to add the new server

To add the new LDAP server into existing replication topology, first create new replication settings such as supplier server (with `ibm-replicasubentry` objectclass) and consumer server (with `ibm-replicationagreement` objectclass)

Example 1. Replication settings for a new peer server

Before you begin, you must know the current existing LDAP servers, their roles, and host names. In this example, there are two LDAP servers with host names of peer1 and peer2. The new peer master's host name is peer3.

With this information, you can build the new settings to add 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 file content

```
#peer master 3 subentry for c=us replication context
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: peer master 3 ibm-replicaSubentry

#peer master 1 to 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: peer master 1 to peer master 3 agreement

#peer master 2 to peer master 3 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: peer master 2 to peer master 3 agreement

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

```
#peer master 3 to peer master 2 agreement
dn: cn=peer2,ibm-replicaServerId=peer3-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: peer master 3 to peer master 2 agreement
```

Example 2. Replication settings for a new replica server

Before you begin, you must know the current existing LDAP servers, their roles, and host names. In this example, there are two existing LDAP servers with host names of peer1 and peer2. The new replica host name is replica3. With this information at hand, you can build the new agreement to be added to the existing enclave.

Create a file called `add_replica_input.txt`, as shown in Example 15-3.

Example 15-3 add_replica_input.txt file content

```
#peer master 1 to peer replica 3 agreement
dn: cn=replica3,ibm-replicaServerId=peer1-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: peer master 1 to peer replica 3 agreement

#peer master 2 to peer master 3 agreement
dn: cn=replica3,ibm-replicaServerId=peer2-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: peer master 1 to peer replica 3 agreement
```

15.2.4 Load the new agreement

To load the new agreement, follow these 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 are using).

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

Note: The `XXXX_input.txt` files are the LDIF files that you already created. They have the necessary new replication settings.

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.

15.2.5 Back up data from a Directory Server v6 peer master server

Now that you have completed the replication setup, synchronize the data from an existing server with the new server. In this section, we describe backup and restore methods. To back up data from a Directory Server v6 peer master server:

1. Log on to one of the Directory Server v6 peer master servers either as root user or as LDAP instance user:
2. Stop the Directory Server v6.

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

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

3. Create a directory to store the LDAP data. Consider a file system with enough free space.

```
# mkdir /home/ldapdb2/dbback
```

4. Change the ownership and permissions on `/opt/tmp/dbback`. The `ldapdb2` and `dbsysadm` group require full access to the `dbback` directory.

```
# chmod 770 /home/ldapdb2/dbback
```

```
# chown ldapdb2:dbsysadm /home/ldapdb2/dbback
```

5. Make a full backup of the existing Directory Server v6 directory.

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

```
# idsdbback -I ldapdb2 -k /opt/tmp/dbback -n
```

6. Compress the data from the Directory Server v6 master server.

```
# cd /home/ldapdb2
```

```
# tar cvf ids6backup.tar dbback
```

7. Compress the tar file. The output file is named `ids6backup.tar.gz`.

```
# gzip ids6backup.tar
```

15.2.6 Restore data to the new LDAP server

Important: Do *not* start any of the Directory Server 6.0 LDAP servers until the following steps are completed on all servers.

To restore data to the new LDAP server:

1. Log on to the new Directory Server v6 server either as root user or as instance user.

2. Transfer or FTP the `ids6backup.tar.gz` file from the Directory Server v6 peer master server `/home/ldapdb2` directory to the `/home/ldapdb2` directory on the new Directory Server v6 server.
3. Extract the Directory Server v6 master server data from the compressed file:


```
# cd /home/ldapdb2
# gunzip ids6backup.tar.gz
```
4. Decompress the `ids6backup.tar` file into the `/home/ldapdb2` directory:


```
# tar xvf ids6backup.tar
```
5. Restore the data into the LDAP database. Using `-r` prevents the `ibmslapd.conf` file from being overwritten by the `ibmslapd.conf` from the Directory Server v6 master server:

Note: The `-I` in the following command is a capital letter *i*.

```
# idsdbrestore -I ldapdb2 -k /home/ldapdb2/dbback -r -n
```

15.2.7 Start all new LDAP servers and verifying replication queues

Important: Start the LDAP servers *only* after you have run `idsdbrestore` on all *new* LDAP servers.

Repeat the steps in this section for each new LDAP server. Start the servers in the following order:

1. Servers with supplier function, such as peer servers
2. Servers with consumer function, such as replica servers

Now complete these steps:

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

Note: Here, too, the `-I` is a capital letter *i*.

```
# idsslapd -I ldapdb2
```

3. Verify that the LDAP server is running:

```
# ldapsearch -p 389 -D cn=root -w passwd -s base -b "" objectclass=*
```

The beginning and the end of the output should look like this:

```
namingcontexts=CN=SCHEMA
. . .
ibm-slapdisconfigurationmode=FALSE
```

4. **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 proceeding to the next step.
5. Verify the replication status by running the following `ldapsearch` against each server that functions as a supplier:

```
# ldapsearch -p 389 -D cn=root -w passwd -s sub -b ""
objectclass=ibm-replicationagreement ibm-replicationstate
```

The beginning and the end of the output should look like this:

```
. . .  
ibm-replicationstate=ready
```

6. Verify that the state is Ready for each replication agreement that leads from the server where you are searching.

15.3 Using the `idsideploy` method to add a new LDAP server

Using the `idsideploy` method, a new directory server instance (target) can be created from an existing directory server instance (source). With this method, configuration, schema, and directory key stash file gets copied into the new server from the old server. The new directory server instance can be configured as a replica or a peer to the source directory server instance if it is in an existing replication deployment.

The `idsideploy` method cannot be used for deploying the first peer or replica server from a stand-alone server. You need at least two LDAP servers in the replication topology to use `idsideploy` to build more LDAP servers. This means that you need to have existing replication topology defined, including at least one replication agreement to one of the existing servers.

Configure an online backup for the source server instance. The backup path must be a shared location, such as NFS, which is common to both the source and target servers.

15.3.1 Prepare an existing peer (source) server

This method requires a shared location for the online backup from the source server, which is used to restore a database on a target server. Consider the `/data` folder as the shared location. It is an NFS mount, and both the source and target systems can read and write to this folder.

1. From the source system, create a folder for backup in a shared folder, and set appropriate ownership:

```
mkdir -p /data/ldapdb2/backup  
chown ldapdb2:idsldap /data/ldapdb2/backup
```

2. Now stop the LDAP server `ibmslapd` process on the source server to make your first offline backup:

```
ibmslapd -I ldapdb2 -k
```

3. On the source server, make an offline backup and enable the online backup (`-u`):

```
idsdbback -I ldapdb2 -k /data/ldapdb2/backup -u -n
```

4. Make sure that the command completed without any problems.

5. Then, start the LDAP server `ibmslapd` process:

```
ibmslapd -I ldapdb2 -n
```

15.3.2 Use `idsideploy` on the second (target) server

Use the `idsadduser` command to create an `ldapdb2` user:

```
idsadduser -u ldapdb2 -w passwd -l /home/ldapdb2 -g idsldap -n
```

This target system must also have access to the `/data` NFS mount.

The **idsideploy** command does the following tasks:

1. Creates an instance.
2. Creates and configures new database.
3. Configure the admin DN and password.
4. Copies configuration and schema from source server instance.
5. Builds and adds necessary replication settings for required combinations.
6. Does an online backup of the source server instance.
7. Restores the backup into the target server instance.

Start the **idsideploy** tool to create a copy of the existing instance on the target server (this command must be run as root user):

```
idsideploy -a db2ldap -I ldapdb2 -e encrypt_seed -D cn=root -w adminpw -sU  
ldap://peer1:389 -sD cn=root -sw adminpw -L /data/ldapdb2/backup -r peer -n
```

Note: To create a replica by using this method, replace peer with replica with the **-r** option.

This command results in the output shown in Example 15-4.

Example 15-4 Output from idsideploy command

```
GLPWRP123I The program '/opt/IBM/ldap/V6.3/sbin/64/idsideploy' is used with the  
following arguments '-a ***** -I ldapdb2 -e ***** -D cn=root -w ***** -sU  
ldap://peer1:389 -sD cn=root -sw ***** -L /data/ldapdb2/backup -r peer -n'.  
You have chosen to perform the following actions:  
  
GLPIDL020I A new directory server instance 'ldapdb2' will be created from the  
source directory server instance, 'ldap://peer1:389'.  
GLPICR057I The directory server instance will be created at: '/home/ldapdb2'.  
GLPICR013I The directory server instance's port will be set to '389'.  
GLPICR014I The directory server instance's secure port will be set to '636'.  
GLPICR015I The directory instance's administration server port will be set to  
'3538'.  
GLPICR016I The directory instance's administration server secure port will be set  
to '3539'.  
GLPICR019I The description will be set to: 'Created from ldap://peer1:4389'.  
GLPICR021I Database instance 'ldapdb2' will be configured.  
GLPICR028I Creating directory server instance: 'ldapdb2'.  
GLPICR025I Registering directory server instance: 'ldapdb2'.  
GLPICR026I Registered directory server instance: : 'ldapdb2'.  
GLPICR049I Creating directories for directory server instance: 'ldapdb2'.  
GLPICR050I Created directories for directory server instance: 'ldapdb2'.  
GLPICR043I Creating key stash files for directory server instance: 'ldapdb2'.  
GLPICR044I Created key stash files for directory server instance: 'ldapdb2'.  
GLPICR040I Creating configuration file for directory server instance: 'ldapdb2'.  
GLPICR041I Created configuration file for directory server instance: 'ldapdb2'.  
GLPICR034I Creating schema files for directory server instance: 'ldapdb2'.  
GLPICR035I Created schema files for directory server instance: 'ldapdb2'.  
GLPICR037I Creating log files for directory server instance: 'ldapdb2'.  
GLPICR038I Created log files for directory server instance: 'ldapdb2'.  
GLPICR088I Configuring log files for directory server instance: 'ldapdb2'.  
GLPICR089I Configured log files for directory server instance: 'ldapdb2'.  
GLPICR085I Configuring schema files for directory server instance: 'ldapdb2'.  
GLPICR086I Configured schema files for directory server instance: 'ldapdb2'.  
GLPICR073I Configuring ports and IP addresses for directory server instance:  
'ldapdb2'.
```

GLPICR074I Configured ports and IP addresses for directory server instance: 'ldapdb2'.

GLPICR077I Configuring key stash files for directory server instance: 'ldapdb2'.

GLPICR078I Configured key stash files for directory server instance: 'ldapdb2'.

GLPICR046I Creating profile scripts for directory server instance: 'ldapdb2'.

GLPICR047I Created profile scripts for directory server instance: 'ldapdb2'.

GLPICR103I Adding instance information to the .profile file for directory server instance: 'ldapdb2'.

GLPICR104I Added instance information to the .profile file for directory server instance: 'ldapdb2'.

GLPICR069I Adding entry to /etc/inittab for the administration server for directory instance: 'ldapdb2'.

GLPICR070I Added entry to /etc/inittab for the administration server for directory instance: 'ldapdb2'.

GLPICR118I Creating runtime executable for directory server instance: 'ldapdb2'.

GLPICR119I Created runtime executable for directory server instance: 'ldapdb2'.

GLPCTL074I Starting admin server for directory server instance: 'ldapdb2'.

GLPCTL075I Started admin server for directory server instance: 'ldapdb2'.

GLPICR029I Created directory server instance: : 'ldapdb2'.

GLPICR031I Adding database instance 'ldapdb2' to directory server instance: 'ldapdb2'.

GLPCTL002I Creating database instance: 'ldapdb2'.

GLPCTL003I Created database instance: 'ldapdb2'.

GLPICR133I Setting the DB2 registry for database instance 'ldapdb2' to allow DB2 SELECTIVITY.

GLPICR134I The DB2 registry for database instance 'ldapdb2' has been set to allow DB2 SELECTIVITY.

GLPCTL017I Cataloging database instance node: 'ldapdb2'.

GLPCTL018I Cataloged database instance node: 'ldapdb2'.

GLPCTL008I Starting database manager for database instance: 'ldapdb2'.

GLPCTL009I Started database manager for database instance: 'ldapdb2'.

GLPCTL049I Adding TCP/IP services to database instance: 'ldapdb2'.

GLPCTL050I Added TCP/IP services to database instance: 'ldapdb2'.

GLPICR081I Configuring database instance 'ldapdb2' for directory server instance: 'ldapdb2'.

GLPICR082I Configured database instance 'ldapdb2' for directory server instance: 'ldapdb2'.

GLPICR052I Creating DB2 instance link for directory server instance: 'ldapdb2'.

GLPICR053I Created DB2 instance link for directory server instance: 'ldapdb2'.

GLPICR032I Added database instance 'ldapdb2' to directory server instance: 'ldapdb2'.

You have chosen to perform the following actions:

GLPCDB023I Database 'ldapdb2' will be configured.

GLPCDB024I Database 'ldapdb2' will be created at '/home/ldapdb2'

GLPCDB035I Adding database 'ldapdb2' to directory server instance: 'ldapdb2'.

GLPCTL017I Cataloging database instance node: 'ldapdb2'.

GLPCTL018I Cataloged database instance node: 'ldapdb2'.

GLPCTL008I Starting database manager for database instance: 'ldapdb2'.

GLPCTL009I Started database manager for database instance: 'ldapdb2'.

GLPCTL026I Creating database: 'ldapdb2'.

GLPCTL027I Created database: 'ldapdb2'.

GLPCTL034I Updating the database: 'ldapdb2'

GLPCTL035I Updated the database: 'ldapdb2'

GLPCTL020I Updating the database manager: 'ldapdb2'.

GLPCTL021I Updated the database manager: 'ldapdb2'.
 GLPCTL023I Enabling multi-page file allocation: 'ldapdb2'.
 GLPCTL024I Enabled multi-page file allocation: 'ldapdb2'.
 GLPCDB005I Configuring database 'ldapdb2' for directory server instance:
 'ldapdb2'.
 GLPCDB006I Configured database 'ldapdb2' for directory server instance: 'ldapdb2'.
 GLPCTL037I Adding local loopback to database: 'ldapdb2'.
 GLPCTL038I Added local loopback to database: 'ldapdb2'.
 GLPCTL011I Stopping database manager for the database instance: 'ldapdb2'.
 GLPCTL012I Stopped database manager for the database instance: 'ldapdb2'.
 GLPCTL008I Starting database manager for database instance: 'ldapdb2'.
 GLPCTL009I Started database manager for database instance: 'ldapdb2'.
 GLPCDB003I Added database 'ldapdb2' to directory server instance: 'ldapdb2'.
 You have chosen to perform the following actions:

GLPDPW004I The directory server administrator DN will be set.
 GLPDPW005I The directory server administrator password will be set.
 GLPDPW009I Setting the directory server administrator DN.
 GLPDPW010I Directory server administrator DN was set.
 GLPDPW006I Setting the directory server administrator password.
 GLPDPW007I Directory server administrator password was set.
 GLPIDL044I Copying audit information from the source server.
 GLPIDL043I Copying password policy information from the source server.
 GLPIDL037W An SSL connection has not been established with the source directory
 server instance; therefore, information about the SSL configuration of that server
 will not be copied to the new directory server instance.
 GLPIDL047I Copying Digest information from the source server.
 GLPIDL045I Copying ulimit information from the source server.
 GLPIDL041W Any additional plugins defined on the source directory server instance
 will not be copied to the target directory server instance configuration file. If
 necessary, plugins must be made available to the target system and added to the
 configuration file before use.
 GLPIDL042I Copying schema information from the source server.
 GLPIDL056I A subentry has been created in the replication topology for the
 directory server ID 'a770b3c0-2991-1031-91e8-eafa5f3b6d12' under the replication
 context 'CN=IBMPOLICIES' with ibm-replicationServerIsMaster set to 'TRUE'.
 GLPIDL057I An agreement has been created in the replication topology from the
 supplier directory server with ID 'f284c9c0-2985-1031-8b22-e2680461574d' to the
 consumer directory server with ID 'a770b3c0-2991-1031-91e8-eafa5f3b6d12' for the
 context 'CN=IBMPOLICIES' using a consumer URL of 'ldap://peer3:389' and the
 credentials DN of 'cn=replcred,cn=replication,cn=ibmpolicies'.
 GLPIDL057I An agreement has been created in the replication topology from the
 supplier directory server with ID 'a770b3c0-2991-1031-91e8-eafa5f3b6d12' to the
 consumer directory server with ID 'f284c9c0-2985-1031-8b22-e2680461574d' for the
 context 'CN=IBMPOLICIES' using a consumer URL of 'ldap://peer1:389' and the
 credentials DN of 'cn=replcred,cn=replication,cn=ibmpolicies'.
 GLPIDL058I The default credentials have been added to the target server's
 configuration file under the DN 'CN=MASTER SERVER,CN=CONFIGURATION'.
 GLPIDL056I A subentry has been created in the replication topology for the
 directory server ID 'a770b3c0-2991-1031-91e8-eafa5f3b6d12' under the replication
 context 'o=sample' with ibm-replicationServerIsMaster set to 'TRUE'.
 GLPIDL057I An agreement has been created in the replication topology from the
 supplier directory server with ID 'f284c9c0-2985-1031-8b22-e2680461574d' to the
 consumer directory server with ID 'a770b3c0-2991-1031-91e8-eafa5f3b6d12' for the

context 'o=sample' using a consumer URL of 'ldap://peer3:389' and the credentials DN of 'cn=repliccred,cn=replication,cn=ibmpolicies'.

GLPIDL057I An agreement has been created in the replication topology from the supplier directory server with ID 'a770b3c0-2991-1031-91e8-eafa5f3b6d12' to the consumer directory server with ID 'f284c9c0-2985-1031-8b22-e2680461574d' for the context 'o=sample' using a consumer URL of 'ldap://peer1:389' and the credentials DN of 'cn=repliccred,cn=replication,cn=ibmpolicies'.

GLPIDL057I An agreement has been created in the replication topology from the supplier directory server with ID 'a770b3c0-2991-1031-91e8-eafa5f3b6d12' to the consumer directory server with ID '646f5eb1-a69c-45c1-95a0-19beb3678c06' for the context 'CN=IBMPOLICIES' using a consumer URL of 'ldap://peer2:389' and the credentials DN of 'cn=repliccred,cn=replication,cn=ibmpolicies'.

GLPIDL057I An agreement has been created in the replication topology from the supplier directory server with ID 'a770b3c0-2991-1031-91e8-eafa5f3b6d12' to the consumer directory server with ID '646f5eb1-a69c-45c1-95a0-19beb3678c06' for the context 'o=sample' using a consumer URL of 'ldap://peer2:389' and the credentials DN of 'cn=repliccred,cn=replication,cn=ibmpolicies'.

GLPIDL048I Replication has been successfully configured. The target source server is now a 'peer' in the replication topology. All agreements that were created have been put on hold and must be resumed for replication to begin.

GLPIDL064I Performing online backup of database on source directory server.

GLPIDL052I The backup of the database for the source directory server instance has been performed and the backed up information was saved in the '/data/ldapdb2/backup' directory.

GLPIDL049I The database for the directory server instance has been loaded/restored using the backup image in the '/data/ldapdb2/backup' directory.

GLPIDL062I The directory server instance 'ldapdb2' has been created.

Now start **ibmslapd**:

```
/opt/IBM/ldap/V6.3/sbin/ibmslapd -I ldapdb2
```

15.4 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 other consumer servers. The entries are deleted when the test is complete (these script files are in the test folder).

1. Log on to the original Directory Server v6 master server to perform the tasks that follow, either as root user or as instance user.

2. Add a test organization to c=US by using a peer master server:

```
# idslsapadd -h <peer1> -p 389 -D cn=root -w <password> -f myTestOrg1.ldif
```

Where myTestOrg1.ldif contains the following:

```
dn: o=myTestOrg1, c=us
o: myTestOrg1
objectclass: top
objectclass: organization
```

3. Search for the test organization on a replica server:

```
# idslsapsearch -h <replica> -p 389 -D cn=root -w <password> -b c=us
o=myTestOrg1
```

The output is similar to this:

```
o=myTestOrg1, c=us
o=myTestOrg1
objectclass=top
objectclass=organization
```

Note: Repeat the following steps (add and search) by using a different number for each peer master.

4. Add one or more test users to o=myTestOrg1, c=US by using different peer master servers:

```
# idsldapadd -h <peerN> -p 389 -D cn=root -w <password> -f myTestPerson<N>.ldif
Where myTestOrg1.ldif contains the following:
dn: cn=myTestPerson1, o=myTestOrg1, c=us
cn: myTestPerson1
sn: myTestPerson1
objectclass: top
objectclass: person
```

5. Search a replica server for the test user that you created in the previous step. If an entry is not returned, wait a few minutes and try again. If there is still no entry, there is a problem with the replication. Fix any problems before continuing:

```
# ldapsearch -h <replica> -p 389 -D cn=root -w <password> -b c=us
cn=myTestPerson<N>
```

Note: myTestPerson<N> corresponds to the file that was used to create the entry, where <N> is equal to 1, 2, 3, 4, or 5.

The output is similar to the this:

```
cn=myTestPerson1, o=myTestOrg1, c=us
cn=myTestPerson1
sn=myPersonSn1
objectclass=top
objectclass=person
```

6. Delete the test entries that were created:

```
# idsldapdelete -h <peer1> -p 389 -D cn=root -w <password> "cn=myTestPerson1,
o=myTestOrg1, c=us"
# idsldapdelete -h <peer1> -p 389 -D cn=root -w <password> "o=myTestOrg1, c=us"
```

If all of the myTestPersons were added, the output will be blank.

7. 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.



Special operating system tuning for IBM Directory Server

In this appendix, we describe specific tuning for an HP-UX, Oracle Solaris, Linux, and IBM AIX operating system (OS), as well as other OS-related environment variables. We cover the following details:

- ▶ Oracle Solaris and HP-UX operating system tuning
- ▶ Linux operating system tuning
- ▶ IBM AIX operating system tuning

Oracle Solaris and HP-UX operating system tuning

The IBM DB2 database system has a tool called *db2osconf* that can make recommendations for kernel parameter values based on the size of a system. The suggested values are high enough that they can accommodate most reasonable workloads. This command is available only for DB2 on HP-UX on 64-bit instances and the Oracle Solaris operating environment.

- ▶ On DB2 for HP-UX, no authorization is required. To make the changes that are recommended by the **db2osconf** utility, you must have root access.
- ▶ On DB2 for the Oracle 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 that are required for DB2 and LDAP:

1. Enter the following command:

```
# /opt/IBM/db2/V9.7/bin/db2osconf -x
```

The results depend on the size of the server and 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 vary 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. Type **y** and press **Enter**.

The output is similar to this:

Back up /etc/system and 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
```

Linux operating system tuning

Before installing an IBM DB2 database system, update your Linux kernel parameters. The default values for particular kernel parameters on Linux are not sufficient when you are running a DB2 database system. DB2 automatically raises the interprocess communication (IPC) limits where necessary, based on the needs of the DB2 database system. However, it might be more practical to set the limits permanently on your Linux system if you have products or applications other than a DB2 database system.

You must have root authority to modify kernel parameters.

To update kernel parameters on Red Hat and SUSE Linux

The following information can be used to determine whether there are any changes that are required for your system.

Example 0-1 depicts the output from the **ipcs -l** command. Comments are added after the **//** to show what the parameter names are.

Example 0-1: Output from the ipcs -l command

```
# ipcs -l
```

```
----- Shared Memory Limits -----
```

```
max number of segments = 4096           // SHMMNI
```

```
max seg size (kbytes) = 32768           // SHMMAX
```

```
max total shared memory (kbytes) = 8388608 // SHMALL
```

```
min seg size (bytes) = 1
```

```
----- Semaphore Limits -----
```

```
max number of arrays = 1024             // SEMMNI
```

```
max semaphores per array = 250          // SEMMSL
```

```
max semaphores system wide = 256000     // SEMMNS
```

```
max ops per semop call = 32             // SEMOPM
```

```
semaphore max value = 32767
```

```
----- Messages: Limits -----
max queues system wide = 1024           // MSGMNI
max size of message (bytes) = 65536      // MSGMAX
default max size of queue (bytes) = 65536 // MSGMNB
```

Now, examine the first section on Shared Memory Limits, specifically the **SHMMAX** and **SHMALL** parameters. **SHMMAX** is the maximum size of a shared memory segment on a Linux system, whereas **SHMALL** is the maximum allocation of shared memory pages on a system.

For **SHMMAX**, the minimum that is required on x86 systems is 268435456 (256 MB) and for 64-bit systems, it is 1073741824 (1 GB).

SHMALL is set to 8 GB by default (8388608 KB = 8 GB). If you have more physical memory than this and it is to be used for DB2, increase this parameter to approximately 90% of the physical memory, as specified for your computer. For instance, if you have a computer system with 16 GB of memory to be used primarily for DB2, then 90% of 16 GB is 14.4 GB divided by 4 KB (the base page size) is 3774873. The **ipcs** output has converted **SHMALL** into kilobytes. The kernel requires this value as a number of pages.

The next section covers the amount of semaphores that are available to the operating system. The kernel parameter semaphore consists of 4 tokens:

- ▶ SEMMSL
- ▶ SEMMNS
- ▶ SEMOPM
- ▶ SEMMNI

SEMMNS is the result of **SEMMSL** multiplied by **SEMMNI**. The database manager requires that the number of arrays (**SEMMNI**) be increased as necessary. Typically, **SEMMNI** should be twice the maximum number of connections allowed (**MAXAGENTS**) multiplied by the number of logical partitions on the database server computer plus the number of local application connections on the database server computer.

The third section covers messages on the system.

MSGMNI affects the number of agents that can be started, **MSGMAX** affects the size of the message that can be sent in a queue, and **MSGMNB** affects the size of the queue.

Change **MSGMAX** to 64 KB (that is, 65535 bytes) and increase **MSGMNB** to 65535 on server systems.

To modify these kernel parameters, edit the `/etc/sysctl.conf` file. Create this file if it does not exist. Example 0-2 shows examples of what to put in the file.

Y Example: 0-2 Parameters to include in the file

```
kernel.sem=250 256000 32 1024
#Example shmmx for a 64-bit system
kernel.shmmx=1073741824
#Example shmall for 90 percent of 16 GB memory
kernel.shmall=3774873
kernel.msgmax=65535
kernel.msgmnb=65535
```

Run **sysctl -p** to load the **sysctl** settings from the default `/etc/sysctl.conf` file.

On a SUSE Linux system, `boot.sysctl` needs to be active to make the changes effective after every reboot. On Red Hat Linux, the `rc.sysinit` initialization script reads the `/etc/sysctl.conf` file automatically.

IBM AIX operating system tuning

This section covers the following performance tuning tasks for the AIX operating system:

- ▶ Enabling large files
- ▶ Setting **MALLOCOPTIONS**
- ▶ 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 that are imposed by the 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, create them as *Enhanced Journaled File Systems* or as *Journaled File Systems with Large File Enabled*. The file system that contains the DB2 instance's home directory and, if bulkload is used, the file system that contains 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 that the maximum file size for that user as unlimited. The soft file size limit can be changed by 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 MALLOCOPTIONS

Malloc buckets provide 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 are 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 by a bucket number. The first bucket is bucket zero, the second is bucket one, the third is bucket two, and so on. The first bucket is the smallest, and each bucket after that is larger than the preceding bucket (using a formula that is described later in this section). A maximum of 128 buckets is available per heap.

Set the **MALLOCOPTIONS** environment variable.

On all AIX 6.x versions, set **MALLOCOPTIONS** by using this command:

```
export MALLOCOPTIONS=buckets
```

If you are using MALLOCOPTIONS buckets, you must set these *ulimits* for the LDAP instance:

```
# ulimit -m unlimited
# ulimit -d unlimited
```

You can find more information about MALLOCOPTIONS in the AIX system documentation.

Setting other environment variables

You might get better performance by tuning the environment variables as shown in the following sections. Check the AIX documentation to see whether these settings might be right for your installation.

AIXTHREAD_SCOPE

On AIX, when using multithreaded applications, especially when running on machines with multiple processors, we strongly advise setting **AIXTHREADSCOPE=S** in the environment before starting the application. This improves performance and produces 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. The **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 you are double-caching unnecessarily.

Suggested values

Generally, the following values can be used for database servers that run on AIX.

- ▶ MINPERM - 10%
- ▶ MAXPERM - 20%
- ▶ MAXCLIENT - 20%

Note: MAXPERM and MAXCLIENT must have *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 that are used for files versus those that are used for computational (working or program text) segments is loosely controlled by the MINPERM and MAXPERM values:

- ▶ If the percentage of RAM that is used 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 the 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 0-3.

Y Example: 0-3 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, the file repaging rate might be higher than the computational repaging rate because the number of file pages in memory is below the MAXPERM value. In that case, you 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%, you can lower the MAXPERM value to 30%. Therefore, the page replacement algorithm steals only file pages. If the **tru_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**):

$$\text{minfree} = 120 * \text{\#cpus}$$
- ▶ Find maxpgahead by using the following command:

$$\text{ioo -a} \mid \text{grep maxpgahead}$$

If maxpgahead is < 8, $\text{maxfree} = 128 * \text{\#cpus}$

If maxpgahead is > 8, $\text{maxfree} = \text{new minfree} + (\text{\#cpus} * \text{maxpgahead})$

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 requesters immediately, without forcing them to wait for page steals and the accompanying I/O to complete. The **MINFREE** limit specifies the free-list size. If you go below that, page stealing to replenish the free list is 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 these conditions are met:

- ▶ Any activity that has critical response-time objectives can always get the page frames that 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, 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 multiprocessor systems with large amounts of RAM. Each memory pool has its own MINFREE and MAXFREE values, but the MINFREE or MAXFREE value 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 0-4 shows a portion of **vmstat** command output on a system where the MINFREE value is being reached.

Y Example: 0-4 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 0-4, 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) is 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 is 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 that are 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 be 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 that are 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 processor use of the AIO servers, if the use is evenly divided among all of them, it 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 best to set maxservers to at least 10*(number of disks that are accessed asynchronously).

This can be achieved for a system with three asynchronously accessed disks:

```
# 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 0-5.

Y Example: 0-5 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/sql1ib
_=/opt/IBM/ldap/V6.0/sbin/64/ibmslapd
instname=ldapdb2
location=/home/ldapdb2
```



How to apply fix packs to an LDAP server

To ensure the health of the IBM Directory Server environment, it is important to stay current with maintenance of the Directory Server and requisite products. To help with this, the IBM Support team maintains a Recommended Fixes document to communicate the fixes and requisite fix levels by release number:

<http://www.ibm.com/support/docview.wss?rs=767&uid=swg27009778>

There are comments and notes that explain important information about known issues and maintenance instructions for Directory Server, IBM DB2 databases, IBM Global Security Kit (GSKit), and IBM WebSphere Application Server software.

There is also an IBM Education Assistant module that shows a step-by-step maintenance application for Directory Server, DB2, and GSKit:

<http://bit.ly/1ihFSSN>



IBM DB2 UDB concepts and definitions

This appendix is a brief summary of some of the IBM DB2 Universal Database™ (UDB) concepts and terminology that are necessary to know to understand the contents of this publication. For a more in-depth discussion of these and other DB2 UDB terms, see the DB2 UDB manuals that are available online, which are listed in “Related publications” on page 241. The best place to go is the IBM DB2 Database for Linux, UNIX, and Windows Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp>

Instances

An *instance* in a DB2 UDB is a logical database manager environment where you can create 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. 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 users. For a partitioned database environment, all database partitions are within a single instance and 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 that contain 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 use and logging.

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 system performance. Data can be accessed much faster from memory than from disk. Therefore, the less 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 that consists of a specific number of columns and a variable number of rows. Tables are uniquely identified units of storage that is 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. To do this, the table must be created in 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
- ▶ Long

In addition to these user-defined tablespaces, DB2 defines separate system and catalog tablespaces. For partitioned database environments, the catalog tablespace is on the catalog database partition.

System-managed versus database-managed tablespaces

For partitioned databases, the tablespaces can be in node groups. During the **CREATE TABLESPACE** command, the containers are assigned to a specific database partition in the node group, so they maintain the *shared nothing* character of DB2 UDB. A tablespace can be either a system-managed space (SMS) or a 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 preallocated file or a physical volume, and the database manager controls the storage space.

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 used only 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. 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. Therefore, it corresponds to the data transfer request size to the disk array.

Before DB2 Version 9.1, the default page size was only 4 KB. Starting with 9.1, the default page size is determined by the table space create statement when the table is created.

Page size determines the maximum length of a row, and it 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

Note:

These are the limits for DMS, SMS, and non-temporary tablespaces. For temporary and automatic storage tablespaces, the limits are much higher. See the “Page, table and table space size” section in the IBM DB2 for Linux, UNIX, and Windows Information Center:

<http://ibm.co/liBp64u>

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. 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 is to *prefetch* can be defined for each tablespace by 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 software. List prefetches are more complex. In this case, the DB2 optimizer optimizes the retrieval of randomly located data. The amount of data that is being prefetched is part of what determines the amount of parallel I/O activity.

Ordinarily, it is best that the database administrator defines 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 by using raw DMS.
- ▶ The tablespace is defined across four containers. Each container is on a separate logical disk, and each logical disk is in a separate Redundant Array of Independent Disks (RAID).
- ▶ The extent size is defined as 16 pages (256 KB).
- ▶ The prefetch value is specified as 64 pages (number of containers x extent size).

Assuming that 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 sent to each container is 16 pages, or 256 KB. The IBM 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 the 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. That causes the array sequential prefetch to take effect.

Page cleaners

Page cleaners write dirty pages from the buffer pool to disk. This reduces the chance that agents that are looking for victim buffer pool slots in memory will incur the cost of writing dirty pages to disk. Waiting for dirty pages to be written to disk is expensive. 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 to disk (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 before their buffer pool memory can be used for other data pages.



DB2 UDB quick reference guide

This appendix lists some of the more important and frequently used IBM DB2 Universal Database (UDB) commands. More information is available on the following websites:

- ▶ IBM DB2 Information Center:
<http://publib.boulder.ibm.com/infocenter/db2help/index.jsp>
- ▶ IBM developerWorks® Information Management section:
<https://www.ibm.com/developerworks/data/>
- ▶ IBM Service Requests (Problem Support 1-800-IBM-SERV):
<http://www.ibm.com/software/support/probsub.html>
- ▶ Fix pack download site for Microsoft Windows, Linux, and UNIX operating systems:
<http://www.ibm.com/support/docview.wss?uid=swg27007053>
- ▶ Fix pack download site for IBM z/OS® systems:
<http://www.ibm.com/support/docview.wss?uid=swg21214640>

DB2 command-line processor

These are among the 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

Use the following for instance configuration:

- ▶ Create: `db2icrt`
- ▶ List: `db2ilist`
- ▶ Update: `db2iupdt`
- ▶ Drop: `db2idrop`

Instance configuration keywords

These are the instance configuration keywords:

- ▶ Display: `GET DBM CFG`
- ▶ Update: `UPDATE DBM CFG USING <keyword> <value>`

For example, to update the instance IP listener configuration, issue this command:

```
UPDATE DBM CFG USING SVCENAME 50000
```

DB2 registry configuration

Use the following for DB2 registry configuration:

- ▶ Help: `db2set -h`
- ▶ Display: `db2set -all`
- ▶ Set: `db2set <variable>=<value>`

For example, to enable the TCP/IP protocol issue this command:

```
db2set DB2COMM=TCPIP
```

Catalog remote database

Use the following for the Configuration Assistance GUI: `db2ca`

```
LIST DB DIRECTORY
LIST NODE DIRECTORY
CATALOG DB <dbname> AT NODE <node name>
CATALOG TCP/IP 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 and stop

These are the DB2 instance start and stop commands:

- ▶ 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

These are the database commands:

- ▶ `CREATE DATABASE <dbname> on <file system path>`
- ▶ `DROP DATABASE <dbname>`
- ▶ `ACTIVATE DB <dbname>`
- ▶ `DEACTIVATE DB <dbname>`

Database connection

Use the following for database connection:

```
CONNECT TO <dbname> [USER] <userid> [USING] <password>
```

- ▶ JDBC T4 String: jdbc:db2://host:port/<dbname>
- ▶ JDBC T2 String: jdbc:db2:<dbname>

Database creation

Example 1. Create a regular DMS table space on a UNIX-based system by using three devices of 10 000 4 K pages each. Specify their I/O characteristics.

```
CREATE TABLESPACE PAYROLL
MANAGED BY DATABASE
USING (DEVICE '/dev/rhdisk6' 10000,
      DEVICE '/dev/rhdisk7' 10000,
      DEVICE '/dev/rhdisk8' 10000)
OVERHEAD 12.67
TRANSFERRATE 0.18
```

Example 2. Create a regular SMS table space on Microsoft Windows NT or Windows 2000, using three directories on three separate drives with a 64-page extent size and a 32-page prefetch size.

```
CREATE TABLESPACE ACCOUNTING
MANAGED BY SYSTEM
USING ('d:\acc_tbsp', 'e:\acc_tbsp', 'f:\acc_tbsp')
EXTENTSIZE 64
PREFETCHSIZE 32
```

Example 3. Create a temporary DMS table space on a UNIX-based system, using two files of 50,000 pages each and a 256-page extent size.

```
CREATE TEMPORARY TABLESPACE TEMPSPACE2
MANAGED BY DATABASE
USING (FILE '/tmp/tempspace2.f1' 50000,
      FILE '/tmp/tempspace2.f2' 50000)
EXTENTSIZE 256
```

Example 4. Create a DMS table space in database partition group ODDNODEGROUP (partitions 1,3, and 5) on a UNIX-based system. Use the /dev/rhdisk0 device for 10,000 4 K pages on each partition. Specify a partition-specific device as 40,000 4 K pages for each partition.

```
CREATE TABLESPACE PLANS
MANAGED BY DATABASE
USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn1hd01' 40000)
ON DBPARTITIONNUM (1)
USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn3hd03' 40000)
ON DBPARTITIONNUM (3)
USING (DEVICE '/dev/rhdisk0' 10000, DEVICE '/dev/rn5hd05' 40000)
ON DBPARTITIONNUM (5)
```

Example 5. Create a regular 9 automatic storage table space named DATATS, and allow the system to make all decisions about table space size and growth.

```
CREATE TABLESPACE DATATS
or
CREATE TABLESPACE DATATS
MANAGED BY AUTOMATIC STORAGE
```

Example 6. Create a regular 9 automatic storage table space named USERSPACE3 with an initial size of 100 megabytes and a maximum size of 1 gigabyte.

```
CREATE TABLESPACE USERSPACE3
    INITIALSIZE 100 M
    MAXSIZE 1 G
```

Example 7. Create a large automatic storage table space named LARGEDATA with a growth rate of 10% (that is, its total size increases by 10% each time that it is automatically resized) and a maximum size of 512 megabytes. Rather than specifying the INITIALSIZE clause, let the database manager determine an appropriate initial size for the table space.

```
CREATE LARGE TABLESPACE LARGEDATA
    INCREASESIZE 10 PERCENT
    MAXSIZE 512 M
```

Example 8. Create a regular DMS table space named USERSPACE4 with two file containers (each container as 1 megabyte in size), a growth rate of 2 megabytes, and a maximum size of 100 megabytes.

```
CREATE TABLESPACE USERSPACE4
    MANAGED BY DATABASE USING (FILE '/db2/file1' 1 M, FILE '/db2/file2' 1 M)
    AUTORESIZE ON
    INCREASESIZE 2 M
    MAXSIZE 100 M
```

Database object display

Use the following to display a 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

Use the following for database configuration:

- ▶ GET DB CFG FOR <dbname>
- ▶ UPDATE DB CFG FOR <dbname> USING <keyword> <value>

Database privileges

Use the following 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

Database statistics updates

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 database monitoring

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

Enable 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

Use the following for troubleshooting:

- ▶ Display DB2 version and service level:
`db2level`
- ▶ Display Java Database Connectivity (JDBC) driver version:
`db2jcc -version`
- ▶ Change diagnostic level for the error message log:
`UPDATE DBM CFG USING DIAGLEVEL 4`
- ▶ Primary error message log file location:
`/INSTHOME/sqllib/db2dump/db2diag.log`



Directory Server backup and restore methods

IBM Security Directory Server uses the IBM DB2 relational database to store directory information. Although most Directory Server administrators might not be interested in the underlying IBM DB2 database structure and how Directory Server uses it, experienced DB2 administrators might be very interested, especially they are when determining backup and restore strategies. This appendix is to help skilled DB2 database administrators design a backup and restore strategy for their own 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, Directory Server). Before considering a backup and restore strategy that includes online backup, be aware that doing an online backup uses a significant amount of DB2 resources. The online procedures are intended only for users with DB2 experience.

For DB2 online backup to be fully supported with Directory Server 6.0, several columns defined in previous Directory Server databases were reduced from a 2 GB to a 1 GB maximum. Although this might seem like a reduction in function, IBM clients' entries are typically less than 24 K, with only a few in the 100 K - 200 K range. As a result, you should be able to make a backup of your Directory Server databases without losing write capabilities.

Beginning with Directory Server Version 6.0, newly created databases are defined with the reduced 1 GB size columns so that they can be defined as a *binary large object*, or *BLOB* (1 G) LOGGED rather than BLOB (2 G) 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 as NOT LOGGED, DB2 does not log the BLOB column. During rollforward recovery, the BLOB is set to NULL. In this case, you might not notice that data is missing until you discover after rollforward recovery that the BLOB columns are NULL. Here, we show how to set up the online backup capabilities in DB2 to use with Directory Server.

For Directory Server 5.2 databases that were migrated to Directory Server 6.0, this section documents DB2 commands that you can use to help you evaluate and, optionally, migrate existing tables for all entries that are less than 1 GB. This migration is *optional* because there

is no easy way in DB2 to drop a column or reduce the size. Therefore, the data must be exported from the existing tables and reloaded into the new tables.

We start with a description of the Directory Server 6.0 database and table space definitions. Individual sections describe alternative Directory Server backup and restore procedures, including DB2 offline and online backup, DB2 offline restore, and redirected restore. If you are migrating data from Directory Server 5.2 to 6.0, the last section provides examples that can be used to evaluate and migrate Directory Server 5.2 databases so that you can use the online backup option.

DB2 information

DB2 backup and restore procedures are described in detail in the *Administration Guides* in the online DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.db2.udb.doc/welcome.htm>

Directory schema and database definitions

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 Directory Server requires backing up the files that contain the Directory Server directory configuration and schema and the DB2 databases.

Directory Server V6.x directory schema

By default, the Directory Server maintains schema files in the Directory Server instance owner's home directory under the etc subdirectory. For example, for the ldapdb2 instance owner, this is the schema file location: /home/ldapdb2/idsslapd-ldapdb2/etc.

You can also specify a different location for the schema files during instance creation.

Each time that 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 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 IBM AIX systems, where ldapdb2 is the Directory Server instance in use and /safeplace/etc is where the schema files are saved, issue the following command:

```
cp /home/ldapdb2/idsslapd-ldapdb2/etc/* /safeplace/etc
```

To set up a new Directory Server with the same schema, issue the following command:

```
cp /safeplace/etc/* /home/ldapdb2/idsslapd-ldapdb2/etc
```

Directory Server V6.x directory database definitions

Because DB2 backup and restore can be done at the database level, the table space level, or both of these levels, it is important to understand the underlying structure to determine which backup and restore method might be best for different Directory Server environments. In general, it is strongly advised that you do *not* do DB2 backup and restore operations at the table space level for reasons that we explain in this section.

In the examples in this paper, `ldapdb2` is the database name. You can use these commands to find the database and table space information for your environment:

```
db2 list database directory
db2 list table space show detail
```

Directory Server directory database and table spaces

When Directory Server creates a database for the directory, it uses the **db2 create database** command to create the database. Directory Server creates this database with four *system-managed space* (SMS) table spaces.

You can view the table spaces by using the following DB2 commands, run within the context of the DB2 instance owner. In this paper, we use the `ldapdb2` user:

```
db2 connect to <databaseName>
db2 list table spaces
```

Example E-1 shows the table space output for the directory database on an AIX, Linux, Oracle Solaris, or HP-UX system.

Example: E-1 Table space 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	

Directory Server data is stored in two separate table spaces: USERSPACE1 and LDAPSPACE. By default, there is only one container or directory for each table space. To view the details about the USERSPACE1 table space, enter a DB2 command similar to this one:

```
db2 list table space containers for 2
```

Example output for the server instance ldapdb2

Example E-2 shows an example of output for the server instance ldapdb2.

Example: E-2 Server output for table space 2

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 table space 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. Table space 3 (LDAPSPACE) contains the remainder of the database tables, which require a 32 K page size. This includes the ldap_entry table, which contains most of the Directory Server directory data and the replication tables. To view the table space container information for the LDAPSPACE table space, enter a DB2 command similar to the following one:

```
db2 list table space containers for 3
```

Example E-3 shows an example of the output.

Example: E-3 Server output for table space 3

Tablespace Containers for Tablespace 3

Container ID	= 0
Name	= /home/ldapdb2/ldap32kcont_1ldapdb2
Type	= Path

It is important to notice that the Directory Server data is spread between table spaces 2 and 3 and that both table spaces must be accessed for most single Directory Server operations. For a search, the attribute tables in table space 2 are used to find the entries that match, but the entry information is returned from the ldap_entry table in table space 3. For an update, the attribute tables in table space 2 must be updated, and the ldap_entry (and possibly the replication tables) in table space 3 must be updated also.

Important: For this reason, it is advisable to back up 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.

Directory Server change log database and table spaces

Directory Server 6.0 has a *change log* function that causes all updates to the directory to be recorded in a separate change log DB2 database (a different database from the one used to hold the 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. Configure the change log function only if you need it, because the additional logging overhead reduces update performance.

One way to check for the status of the change log function is to look for the suffix CN=CHANGELOG. If that exists, the change log function is enabled.

When Directory Server creates a database for the change log, it uses the **db2 create database** command to create a database named *ldapclog*. It creates this database with four system-managed space (SMS) table spaces that are identical to the *ldapdb2* database described.

You can view the table spaces by using the following DB2 commands, run within the context of the DB2 instance owner (the *ldapdb2* user in these examples):

```
db2 connect to ldapclog
db2 list table spaces
```

It is important to notice that the Directory Server Directory information is stored in a different database (*ldapdb2*) from the change log database (*ldapclog*). To keep related sets of data together, 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 Directory Server database across multiple disks.

In some cases, the Directory Server or DB2 administrator might have altered the 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, regardless of whether 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 that follow.

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 where the database is to be distributed.

Follow these guidelines:

- ▶ Because DB2 distributes the database equally across all directories, it is a good idea to make all of the file systems or directories, or both, the same size.
- ▶ All directories to be used for the DB2 database must be empty. AIX and Solaris systems create a `lost+found` directory at the root of any file system. Rather than deleting the `lost+found` directory, create a subdirectory at the root of each file system to use for distributing the database. Create a subdirectory named `diskn` for each file system where the DB2 database is to be stored (for example, `disk1`, `disk2`, `disk3`, and so on).
- ▶ Create two more directories under the `diskn` directory: One for holding table space 2 and the other for table space 3. For example, these directories might be named `ts2` and `ts3`. Then, specify these directories in the **set table space** commands, as explained in “Performing a redirected restore of the database” on page 193.
- ▶ The DB2 instance user must have *write* permission on the created directories. For IBM AIX and Oracle Solaris systems, the following command gives the correct permissions:

```
chown ldapdb2 directory_name
```

Platform-specific guidelines

Keep these file system guidelines in mind:

- ▶ 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. You can find this option 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, follow these steps:

1. Stop the IBM Directory Server process (`ibmslapd`).
2. Enter the following to close all DB2 connections:

```
db2 force applications all
db2 list applications
```

A message similar to the following is displayed:

```
SQL1611W No data was returned by the Database System Monitor.
```

3. Enter the following to initiate the offline backup:

```
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 time stamp for this backup image is: 20000420204056
```


Follow these steps for a redirected restore:

1. Enter the following command to start the DB2 restore process:

```
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 table space containers are  
inAccessible, or has set their state to 'storage must be defined'.
```

```
DB20000I The RESTORE DATABASE command completed successfully.
```

2. The SQL1277N message indicates that *storage must be defined* for the table space containers. To define the containers for table space 2 and for table space 3, enter the commands shown in Example E-5.

Example: E-5 Define the containers for table space 2 and for table space 3

```
db2 "set table space containers for 2 using (path \  
'/disk1/ts2', path '/disk2/ts2', path '/disk3/ts2', \  
path '/disk4/ts2', path '/disk5/ts2')"  
db2 "set table space 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 it within the current shell by 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 table space containers** command, a message similar to the following is displayed:

```
DB20000I The SET TABLESPACE CONTAINERS command completed successfully.
```

If you receive a message that says 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`. Create a directory at the same level as `lost+found` to hold the table space, and then reissue the `set table space` command. If you experience problems, see the DB2 documentation. The following files might also be helpful:

- ▶ `ldapdb2_home_dir /sql1lib/Readme/en_US/Release.Notes`
- ▶ `ldapdb2_home_dir /sql1lib/db2dump/db2diag.log`

The `db2diag.log` file contains some fairly low-level details that can be difficult to interpret. In many cases, the information can be used to determine whether and what type of error occurred.

3. Continue to restore to the new table space containers. This step, which restores the entire database from its previous containers to the new containers, takes the most time to complete. The time varies depending on the size of the directory. Enter the following command to continue the restore the data in the new table space containers:

```
db2 restore db ldapdb2 continue
```

If successful, the following message displays:

```
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 was stopped, the redirected restore can be restarted, beginning at Step 1.

Overview of offline backup and restore procedures for LDAP

The fastest way to back up and restore the database is to use **db2 backup** and **restore** commands. Directory Server alternatives, such as **db2ldif** and **ldif2db**, are usually much slower. Directory Server supports **idsdbback** and **idsdbrestore**, which use the **db2 backup** and **restore** commands and save additional Directory Server configuration and schema information. However, it is important to note that **idsdbback** does not support DB2 *online* backup in Directory Server 6.0. You can use **idsbback** only when the 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 Directory Server and then restore that database on another version of Directory Server. 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 Directory Server **db2ldif** and **ldif2db** commands.

If you restore over an existing database, any performance-tuning tasks on that existing database are lost. Check all DB2 configuration parameters after 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.

Example E-6 on page 196 shows the DB2 commands that you would use for offline backup and restore operations for a directory database named `ldapdb2`, where *directory_or_device* is the name of a directory or device where the backup is stored.

Example: E-6 DB2 commands for offline backup and restore for ldapdb2 database

```
db2 force applications all
db2 backup db ldapdb2 to directory_or_device
db2 restore db ldapdb2 from directory_or_device replace existing
```

Example E-7 shows the DB2 commands for offline backup and restore operations for the change log (ldapclog), where *directory_or_device* is the name of a directory or device where the backup is stored.

Example: E-7 DB2 commands for offline backup and restore for the change log

```
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 correct directories on the new system. If you are 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 191.

Replication considerations

You can use backup and restore operations to initially synchronize a Directory Server consumer with a 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 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, rollforward recovery is not possible; only crash recovery is enabled. The directory server must be stopped and offline when backups are made.

When log archiving is configured for the database, rollforward recovery is possible. This is because the logs record changes to the database after the time that the backup was made. You archive logs by setting the logretain database configuration parameter to RECOVERY. When this parameter is configured, the database is enabled for rollforward 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 whether

the database is in a Backup Pending state, look at the Backup Pending value (YES or NO) that is 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. Rollforward recovery reapplies the completed units of work that are recorded in the logs to the restored database, table space, or table spaces. You can specify rollforward recovery to be either to the end of the logs or to a particular day and time.

A recovery history file is created with each database. The recovery history file is updated automatically with summary information whenever you back up or restore a full database or table space. 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 table space
- ▶ Restoration of a database or table space
- ▶ Rollforward of a database or table space
- ▶ Alteration of a table space
- ▶ Quiescence of a table space
- ▶ Rename of a table space
- ▶ 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 **logretain** and other parameters that are related to rollforward recovery. Because the default parameter settings do not work well in some cases, 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 that 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 rollforward 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 make a full database backup to have a baseline for incremental backups.

Offline and online backup and restore examples

Basic examples for both offline and online backup of the database are described in the following sections. Example E-8 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 sample output for the DB2 backup history.

Example: E-8 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 of an offline backup and restore procedures

The first four steps explain 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. The `/safeplace` directory is used for the examples that follow. (The DB2 instance owner must have *write* permission for the `/safeplace` directory.)

2. Save the Directory Server configuration and schema files in a secure location. These files must be updated only if you change the topology, change your configuration parameters, or change your schema. For this example, 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 next five steps restore the directory database on a different machine:

1. If necessary, install Directory Server.
2. Configure a new database and use the same information that was specified on the backup machine.
3. Copy or use FTP to move 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 Directory Server perspective, you cannot back up a database on one version of Directory Server and restore that database on another version of Directory Server. It is advisable to use the same version of **db2 backup** and **db2 restore** for both DB2 operations.

Example of an online backup for the directory database

The first seven steps describe how to use full online backups 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. The /safeplace directory is used for the examples that follow. (The DB2 instance owner must have write permission for /safeplace.)
2. Save the Directory Server configuration and schema files in a secure location. These files must be updated only if you change the topology, change your configuration parameters, or change your schema. For this example, we use ldapdb2 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. For recovery purposes, keep log files on a different physical drive from the database. For this example, /safeplace/db2logs-ldapdb2 is the secure location. Run this DB2 command 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
```

Note: 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 next two steps shows you how to create nightly full online backups for the directory database:

1. On a nightly basis (or more frequently, if 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"
```

Example E-9 shows an example of a response.

Example: E-9 Example of the information that is returned

Path to log files	= /safeplace/db2logs-ldapdb2/NODE0000/
-------------------	----------------------------------------

Incremental directory and change log database online backup

Follow these steps 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. We use /safeplace for the examples that follow. (If the change log is not configured, all commands containing **ldapclog** can be ignored.)

2. Save the Directory Server configuration and schema files in a secure location:

```
cp /home/ldapdb2/idsslapd-ldapdb2/etc/* /safeplace/etc
```

These files must be updated only if you change the topology, change your configuration parameters, or change your schema. We use ldapdb2 as the Directory Server instance.

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, keep log files 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):

```
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
```

Note: Setting **trackmod on** for incremental backup support can have an impact on the runtime performance for operations that update or insert data.

Creating full offline backups for directory and change log databases

Use these actions 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 you determine that it is necessary), create incremental backups. Run the same commands on Tuesday, Wednesday, Thursday, Friday, and Saturday:

```
db2 backup db ldapdb2 online incremental to /safeplace/mon-ldapdb2
```

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.

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 must be saved in different directories *if* the change log is configured.

2. Verify the path to the log files for the directory database:

```
db2 get db config for ldapdb2 | grep "Path to log files"
```

Example E-10 on page 203 shows the information that is returned by this command.

Example: E-10 Directory database path to log files command results

```
Path to log files    = /home/ldapdb2/ldapdb2/NODE0000/SQL00001/SQLLOGDIR/  
cp /home/ldapdb2/ldapdb2/NODE0000/SQL00001/SQLLOGDIR/*/safeplace/db2logs-ldapdb2  
db2 backup db ldaplog online incremental to /safeplace/mon-ldaplog
```

3. Verify the path to the log files for the change log database:

```
db2 get db config for ldaplog | grep "Path to log files"
```

Example E-11 shows the information returned by this command.

Example: E-11 Change log database path to log files command results

```
Path to log files    = /home/ldapdb2/ldapdb2/NODE0000/SQL00002/SQLLOGDIR/  
cp /home/ldapdb2/ldapdb2/NODE0000/SQL00002/SQLLOGDIR/* /safeplace/db2logs-ldaplog
```

Restoring the directory database

Assume that a disk drive failed on Wednesday morning on the machine that was used in the previous example. Because /safeplace that was used to back up the files and logs was not affected, it can be used for the 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 that are being used and the log files in the /safeplace/db2log-ldapdb2/NODE0000 directory.

1. If necessary, install Directory Server.
2. Configure a new database by using the same information that was specified previously.
3. Copy (or compress) the configuration and schema files that you backed up previously:

```
cp /safeplace/etc/* /home/ldapdb2/idsslapd-ldapdb2/etc
```
4. Restore the directory database from Tuesday night:

Note: The timestamp_of_backup clause is required only if there is more than one backup image in the specified directory path.

```
db2 restore db ldapdb2 from /safeplace/tues-ldapdb2 taken at timestamp_of_backup
```

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 that was 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 one:

```
db2 update db config for ldapdb2 using newlogpath /safeplace/db2logs-ldapdb2
```

6. Roll forward all logs that are 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 that span the time since that backup was made.

Restoring both directory and change log databases

Assume that a disk drive failed on Wednesday morning on the machine that was being used in the previous example. Because the /safeplace directory used to back up the files was not affected, it can be used to 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 that are being used and the log files in these directories:

```
/safeplace/db2log-ldapdb2/NODE0000  
/safeplace/db2log-ldapclog/NODE0000
```

1. If necessary, install Directory Server.
2. Configure a new database, using the same information that was specified previously.
3. Copy the configuration and schema files that you backed up:

```
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, to restore Tuesday's incremental backup, use these commands:

```
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 that you backed up to the default log path locations:

```
cp /safeplace/db2logs-ldapdb2/*/home/ldapdb2/ldapdb2/NODE0000/SQL00001/SQLLOGDIR  
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 that you backed up to the default log path locations:

```
cp /safeplace/db2logs-ldapclog/*  
/home/ldapdb2/ldapdb2/NODE0000/SQL00002/SQLLOGDIR  
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. The Monday incremental backup is not needed to restore through Tuesday.

Using incremental delta backups

In these examples of 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 that are saved for Saturday than there were for Monday. DB2 also allows *delta* backups, which save only changes that were made since the last backup of any kind. These delta backups are much smaller and finish faster. However, when you use them to restore, you will need *all* delta backups made since the last full or incremental backup.

Example E-12 shows commands for nightly online delta backups for this ldapdb2 database.

Example: E-12 Commands for nightly delta backups

```
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 also be kept in a secure place when you use 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 that you are using for restoring the delta backups. If you are using the default log paths, you have two options:

- ▶ Copy them from this directory on the backup machine to the default log path on the machine that is being restored:
/safeplace/db2logs-ldapdb2/NODE0000
- ▶ Or modify the newlogpath database configuration on the new machine and copy them directly to this location:
/safeplace/db2logs-ldapdb2/NODE000

When restoring from delta backups, you must have *all* delta backups that were made since the last full or incremental backup.

Example E-13 shows commands to restore online delta backups for this ldapdb2 database.

Example: E-13 Commands to restore online delta backups

```
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 rollforward as before:

```
cp /safeplace/db2logs-ldapdb2/*  
/home/ldapdb2/ldapdb2/NODE0000/SQL0001/SQLLOGDIR/  
db2 rollforward db ldapdb2 to end of logs and stop
```

Pros and cons of different recovery strategies

If a database has high write activity, a *full online backup* might be more efficient. Although it is 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 *noncumulative backup image*. Although delta backups are smaller, *all* delta backups since the last full or incremental backup are required to restore the database.

Other backup, restore, and rollforward 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 that are made after that time (including *without rolling forward* prevents DB2 from putting the restored database in *rollforward pending* state):

```
db2 restore db ldapdb2 from /safeplace taken at 20040405154705 without rolling  
forward
```

Use this command 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
```

To restore an offline rollforward database to a point in time, use the following command:

```
db2 "rollforward database ldapdb2 to 2004-04-22-14.54.21.253422 and stop
```

This command rolls forward all logs that are located in the log folder that is specified in the database configuration file, up to and including the stated point in time. The *and stop* key phrase completes the rollforward recovery process by rolling back incomplete transactions and tuning off the *rollforward pending* state of the database.

Common problems for backup, restore, rollforward commands

We use `ldapdb2` as the database name in the examples that follow. For change log, the change log database (`ldapclog`) can be used.

Example E-14 shows trying to update database configuration for online backup parameters while `ibmslapd` is running.

Example: E-14 Trying to update database configuration 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 E-15 shows trying to perform an online backup without setting `logretain`.

Example: E-15 Online backup attempt 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 rollforward recovery for database `ldapdb2`, you must run the following DB2 command:

```
db2 update db config for ldapdb2 using logretain recovery
```

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. 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 that the Directory Server fails when it connects to the database, and it starts in configuration mode only.

To do a full backup, use this command:

```
db2 backup database ldapdb2 to /safeplace
```

If the backup is successful, a message such as the following is displayed:

```
Backup successful.
```

The time stamp for this backup image is: 20040308170601

If you try to restore a database when ibmslapd is running, the following message is displayed:
SQL1035N The database is currently in use.

If a roll forward action must be done following a restore, you receive the following message:
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 rollforward pending state until a **rollforward** command is performed. This means that the Directory Server fails when it connects to the database, and it starts in configuration mode only.

Optional migration from V5.2 to V6.0 to enable online backup

This section provides documentation examples that can be used to migrate Directory Server Version 5.2 databases to Version 6.0 so that you can use the online backup option that is available in the later version. 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 resources. This might not be necessary if you do not require online backup and point-in-time recovery.

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). For these examples, we use ldapdb2 and ldapclog as the database names.

In Directory Server 5.2 databases, there are four columns defined as BLOB (2 G) NOT LOGGED NOT COMPACT, as shown in Table E-1.

Table E-1 Directory Server v5.2 BLOB (2 G) 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 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, the data fits 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, do not use online backup unless all BLOB columns are reduced to 1 GB.

Evaluating BLOB columns on Directory Server 5.2 and 6.0

The *blobinfo* example script in Example E-16 can be used to analyze Directory Server tables and determine information for the columns that contain BLOB definitions for a specific user's directory information. It provides detailed information for the number of entries that are 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 *blobmigrate 1G* script. Those entries must be either modified to be less than 1 GB or removed before the *blobmigrate1G* example script can be successfully run.

Example: E-16 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

```

[illegible]

Example: E-17 Directory database that contains no entries with BLOBs greater than 1 GB

Thu Jan 20 11:56:47 GMT 2005

LDAPDB2 Database

Table	Total# Entries	Total# VarChar	Blob Name	Blob Defn*	Total# BLOB	Total# BLOB
					<1G	>1G**
LDAP_ENTRY	1000015	1000015	ENTRYBLOB	2G*	0	0

** There are NO BLOBs > 1G in the LDAPDB2 Database

Example: E-18 Directory database that contains one entry with a BLOB greater than 1 GB

Thu Jan 20 12:56:47 GMT 2005

ldapdb2 Database

Table	Total# Entries	Total# VarChar	Blob Name	Blob Defn*	Total# BLOB	Total# 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 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-19 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# Entries	Total# VarChar	Blob Name	Blob Defn*	Total# BLOB	Total# BLOB
					<1G	>1G**
LDAP_ENTRY	1000015	1000015	ENTRYBLOB	2G*	0	0

** There are NO BLOBs > 1G in the LDAPDB2 Database

LDAPCLOG Database

Table	Total# Entries	Total# VarChar	Blob Name	Blob Defn*	Total# BLOB	Total# 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, notice the following entries:

- ▶ In Example E-17 on page 213, 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-18 on page 213, the entry with a BLOB greater than 1 GB must be deleted before the blobmigrate1G script can be run.
- ▶ In Example E-19 on page 213, the blobmigrate1G script can be run because there are *no* entries with blobs greater than 1 GB.

blobmigrate1G script

The blobmigrate1G script is an example script that can be used to *optionally migrate* migrated V5.2 tables for all entries that are less than or equal to 1 GB. 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 IBM AIX for journaled file system (JFS) or JFS2 use

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. **Tip:** Choose this option when you create the file system.

See the IBM AIX system documentation for additional information and file system options.

Example E-20 shows an example of the sample blobmigrate1G script.

Example: E-20 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 [ $numparam -ne 1 ]
then
    print "usage: blobmigrate1G <output directory>"
    print "NOTES: 1) Script must be run as ldap instance owner"
    print "        2) TDS V6.0 Migration must be DONE"
    print "        3) ibmslapd must NOT be running"
    print "        4) logretain should be turned OFF"
    print "        5) blobs > 1G can NOT be migrated"
    print "        6) databases ldapdb2 and ldapclog must NOT be in use"
    print ""
    print " The summary report is <output directory> blobmigrate1G.rpt"
    exit
fi
#####
. /home/$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 [ "$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,
```



```

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"$totblobsmall"\t"$to
tblobbig >> $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

```



```

totblobbig=`cat $outdir/$ldapclog.ldapentry.beforecounts | awk '{if (NR==18)
print $1}'`
print
totblobbig >> $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 [ "$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" >>
$outdir/blobmigrate1G.rpt
fi
diffclog=""
if [ "X$chgllog" != "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 $ldapdb2
    if [ "X$chgllog" != "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-21 shows sample blobmigrate1G.rpt output.

Example: E-21 Directory database that contains no entries with BLOBs greater than 1 GB

```

./blobmigrate1G /outdir
Begin backup database ldapdb2 to /outdir at:
Thu Jan 20 13:11:11 GMT 2005
Backup successful. The time stamp 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# Entries	Total# VarChar	Blob Defn	Total# BLOB<1G	Total# 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

```



Checklists

In this appendix, we provide several checklists that cover maintenance, monitoring, and other Lightweight Directory Access Protocol (LDAP) utilities. We describe details of these two checklists:

- ▶ Maintenance checklist
- ▶ Monitoring checklist

Maintenance checklist

We recommend the following IBM Directory Server maintenance steps:

- ▶ Back up the Directory Server daily.
See Appendix E, “Directory Server backup and restore methods” on page 187 for details:
 - Database backup
 - Directory Server important files backup
 - Backup of the current day’s changes files
- ▶ Rotate and examine 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 an error, but you might not have to rotate the other log files.

For details in problem determination, see *IBM Tivoli Directory Server Problem Determination Guide Version 6.0*, SC32-1679.

- ▶ Check whether the hard disk has enough space for the log files and database to grow.
- ▶ Check performance every month. Tune the Directory Server if necessary.
- ▶ Monitor Directory Server status by using either the web administration tool or command-line utilities.
- ▶ Prepare Directory Server failover procedures.
- ▶ Prepare Directory Server backup and restore procedures.
- ▶ Check for the latest Directory Server Fix Pack or upgrade.

IBM Security Directory Server support tool

The support tool collects relevant data, such as logs, directory listings, schema files, and core files. The tool then packages the information into a compressed file archive that you can send to IBM Support for help in diagnosing your problem. The *IBM Tivoli Directory Server Problem Determination Guide Version 6.0*, SC32-1679, describes the information that is collected by the support tool and includes instructions for generating the idssupport file.

Monitoring checklist

Monitoring Directory Server for problems can be broken down into the following categories:

- ▶ Monitoring for outages or performance degradations:
The Directory Server consists of two primary processes that must be active and using resources for Directory Server to be healthy:
 - The LDAP server process (unique name that reflects the Directory Server instance name)
 - UDB or IBM DB2 database
- ▶ Monitoring for performance:
 - auditmonit.jar file
 - **ldapsearch** utility

- ▶ Monitor Directory Server status by using either the web administration tool or command-line utilities, such as **ldapsearch** or the IBM Tivoli Directory Integrator monitoring utility.
- ▶ Analyze the Directory Server log files to check the status of periodically.
- ▶ Monitoring Directory Server performance.
- ▶ Optional:
You can also set up IBM Security Directory Integrator assembly lines to monitor Directory Server status and send an automated email message to the system administrator when Directory Server fails. Additional effort is necessary to build that function in IBM Tivoli Directory Integrator.
- ▶ Optional:
You can also use other IBM Tivoli professional monitoring products, although that requires additional effort to set up such service.

Key reasons for monitoring

As for any application, the ability of an administrator to understand what the current state of the application is at any particular time is critical. Monitoring of the directory is important from the following perspectives:

Security issues

To track unauthorized access and take corrective measures. For this example, we want to prevent unauthorized access to the directory. Monitoring the server helps overcome issues such as these and, consequently, secures our server.

Performance issues

Slow or poor performance of the Directory Server can be for many reasons:

- ▶ DB2 buffer pools might not be tuned according to the directory requirements.
- ▶ Directory Server caches might not be tuned optimally.
- ▶ Important indexes might be missing.
- ▶ Database reorganization might be required.

Some of these problems might be caught or prevented by using the monitoring tools. Monitoring tools help you derive 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 specified time frame, how many additions have been completed, how many binds to the server have occurred, and how many operations have been completed. Keeping track of such figures helps you calculate the throughput of the server. This throughput might be influential for the dependent products that use the Directory Server.

Anomalies

The Directory Server includes a set of tools to monitor any anomalies in the following ways:

- ▶ Searching against the `cn=monitor` base
- ▶ Searching through the change log database
- ▶ Analyzing log files

Next, we explain some of the details involved in monitoring the Directory Server.

Monitoring for outages and performance degradations

The Directory Server consists of two primary processes that must be active and using resources for Directory Server to be healthy:

- ▶ The LDAP server process (unique name that reflects the Directory Server instance name)
- ▶ UDB or 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 stops, the monitoring tool should attempt at least one iteration of an automated restart and log the event.
- ▶ Monitoring log files:

These Directory Server 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 these:

- Excessive failures of root account binds
- Abandoned connections (usually due to a client, network, or Directory Server not responding in time)

- ▶ Query of the Directory Server `cn=monitor` object:

The `cn=monitor` object should be read regularly (for example, once a minute) to determine possible error conditions, for example:

- Emergency thread active
- Depth of work queue
- `OpsCompleted` is incrementing (when client applications are not in maintenance or 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 file 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 and DB2 process

Monitoring the UDB and DB2 process can be broken down into three distinct areas:

- ▶ The UDB or DB2 server process must be active and in memory. If this process stops, it should be automatically restarted as it is normally configured in the `inittab`. The monitoring tool that is chosen should at least log the event and send an alert if the process is not restarted within 30 seconds.

- ▶ Monitoring log files:

These critical DB2 log files must be monitored for operational issues:

- `Idsldap.nfy`:

A shortened version of the `db2diag.log` file that contains only errors.

- Correct startup
- Error conditions, such as out of space, and so on

- `db2diag.log`:

Monitor this log file 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 performance and service level agreement conformance

The best method to determine the average performance of operations is using synthetic transactions (caveat: advisable only for large directories). 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 time. These are the critical factors in the development of these synthetic transactions:

- ▶ Run them on the same server as the directory (or proxy) to ensure that network latency is not an issue.
- ▶ Run them randomly to ensure that caches do not skew the data.
- ▶ Ensure that they are lightweight (both the operations and the initiating application) so that they do not affect the production performance of the directory.

Monitoring Directory Server status

You can monitor Directory Server status by using either the web administration tool or command-line utilities, such as the LDAP search utility.

Operating system command line to monitor Directory Server

Run this command-line utility to view information about the running `ibmslapd` process:

```
ps -eaf | grep -i ibmslapd
```

This command helps you monitor two things:

- ▶ Whether `ibmslapd` is still running.
- ▶ How much the process size has grown until this date and whether it is within permissible limits or going to reach the limits soon.

Web administration tool

You can use the web administration tool for several tasks:

View server status

You must log in to the web administration tool to view the server status, as shown in Figure F-1.

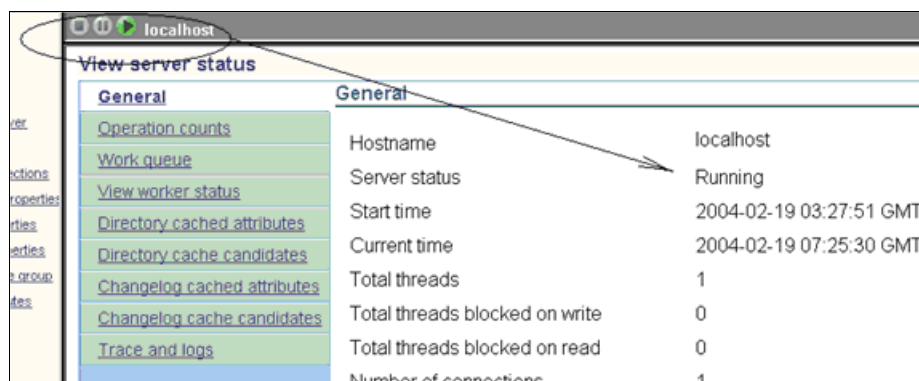


Figure F-1 Viewing the server status

View the worker thread status

The state of a worker thread includes many details, such as thread number, information about the client that it is serving, the type of work request received, and so on. Performing this activity suspends all server activity until it is completed. See Figure F-2 and Figure F-3.

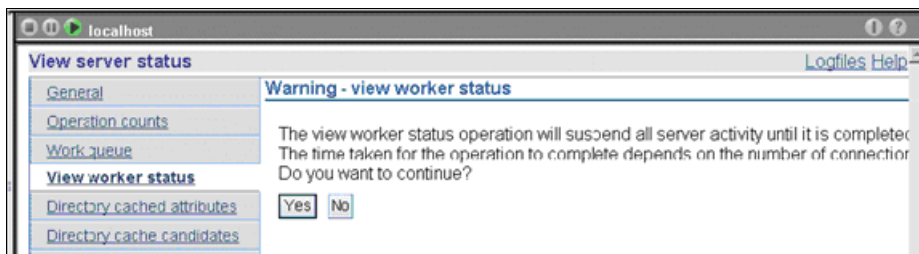


Figure F-2 Viewing the worker thread status

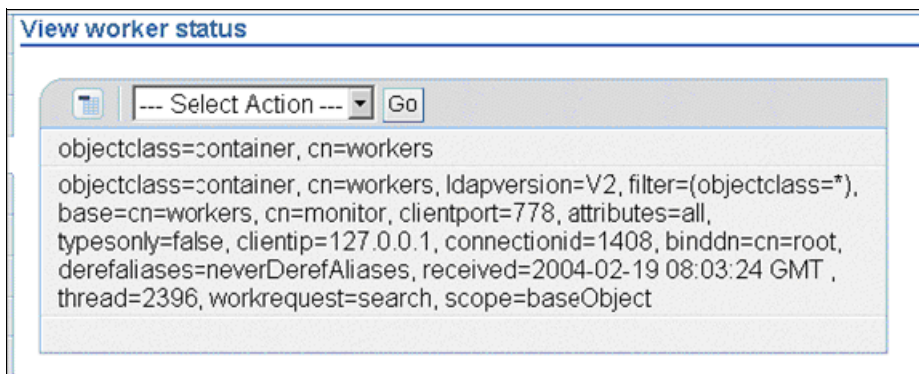


Figure F-3 Viewing the worker thread status (continued)

View connections

See Figure F-4 on page 231.

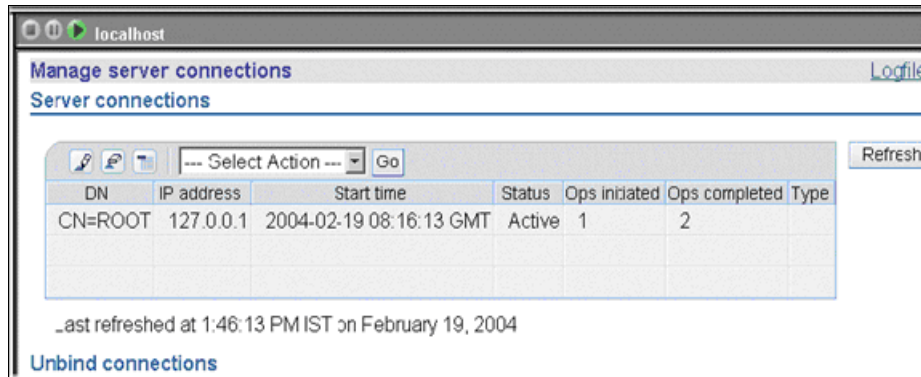


Figure F-4 View connections

View other server status

1. Connect to the Directory Server by using the web administration tool.
2. Click **View server status**. This panel has nine tabs.

Command-line

You can use the command line for the following purposes:

View server status

The following command returns the state of the server:

```
ibmdirctl-D <adminDN> -w <adminPW> status
```

This command does not say whether the server is running in safe mode or not. Confirm it by running a **rootDSE** search after this command, and check this attribute, which it should be false for a normal mode:

```
ibm-slapdisconfigurationmode
```

On UNIX:

```
ldapsearch -D <adminDN> -w <adminPW> -s base objectclass=* | grep config
```

View worker thread status

View worker thread status by using the following command lines.

To retrieve all information that is 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=*
```

The output is similar to what Example F-1 shows.

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

You can run a search with the search base of `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 `ldapsearch` against base `cn=monitor`. The command for a monitor search is:

```
ldapsearch -D <adminDN> -w <adminPW> -s base -b cn=monitor objectclass=*
```

Using the `ldapsearch` utility for monitoring

The `ldapsearch` command can be used to monitor performance, as shown in the following sections.

The `cn=monitor` command now extracts more data to better monitor how the Directory Server is doing. The monitor search returns the following attributes of the server:

cn=monitor version:	IBM Tivoli Directory, Version 5.2
total connections:	Number of connections since the server was started
current connections:	Number of active connections
maxconnections:	Maximum number of active connections allowed
writewaiters:	Number of threads that are sending data back to the client

readwaiters:	Number of threads that are reading data from the client
opsinitiated:	Number of initiated requests since the server was started
livethreads:	Number of worker threads that are being used by the server
opscompleted:	Number of completed requests since the server was started
entriessent:	Number of entries sent by the server since it was started
searchesrequested:	Number of initiated searches since the server was started
searchescompleted:	Number of completed searches since the server was started
filter_cache_size:	Maximum number of filters that are allowed in the cache
filter_cache_current:	Number of filters currently in the cache
filter_cache_click:	Number of filters that are retrieved from the cache rather than being resolved in DB2
filter_cache_miss:	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:	Maximum number of entries that are allowed in the cache
entry_cache_current:	Number of entries currently in the cache
entry_cache_click:	Number of entries that were retrieved from the cache
entry_cache_miss:	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 access control list (ACL) cache is active (TRUE) or inactive (FALSE)
acl_cache_size:	Maximum number of entries in the ACL cache
currenttime:	Current time on the server. The current time is in the format: year month day hour:minutes:seconds GMT.
starttime:	Time the server was started. The start time is in the format: year month day hour:minutes:seconds GMT.
en_currentregs:	Current number of client registrations for event notification
en_notificationssent:	Total number of event notifications sent to clients since the server was started

The following attributes are used for operation counts:

bindsrequested:	Number of bind operations requested since the server started
bindscompleted:	Number of bind operations completed since the server started
unbindsrequested:	Number of unbind operations requested since the server was started
unbindscompleted:	Number of unbind operations completed since the server was started
addsrequested:	Number of add operations requested since the server was started
addscompleted:	Number of add operations completed since the server was started
deletesrequested:	Number of delete operations requested since the server was started

deletescompleted:	Number of delete operations completed since the server was started
modrdnsrequested:	Number of modify relative distinguished name (RDN) operations requested since the server was started
modrdnscompleted:	Number of modify RDN operations completed since the server was started
modifiesrequested:	Number of modify operations requested since the server was started
modifiescompleted:	Number of modify operations completed since the server was started
comparesrequested:	Number of compare operations requested since the server was started
comparescompleted:	Number of compare operations completed since the server was started
abandonsrequested:	Number of abandon operations requested since the server was started
abandonscompleted:	Number of abandon operations completed since the server was started
extopsrequested:	Number of extended operations requested since the server was started
extopscompleted:	Number of extended operations completed since the server was started
unknownopsrequested:	Number of unknown operations requested since the server was started
unknownopscompleted:	Number of unknown operations completed since the server was started

The following attributes are used for server logging counts:

slapderrorlog_messages:	Number of server error messages recorded since the server was started or since a reset was performed
slapdclierrors_messages:	Number of DB2 error messages recorded since the server was started or since a reset was performed
auditlog_messages:	Number of audit messages recorded since the server was started or since a reset was performed
auditlog_failedop_messages:	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:	Total number of Secure Sockets Layer (SSL) connections since the server was started
total_tls_connections:	Total number of Transport Layer Security (TLS) connections since the server was started

The following attributes are used for tracing:

trace_enabled:	Trace value for the server (TRUE if collecting trace data, FALSE if not collecting trace data)
-----------------------	------------------------------------------------------------------------------------------------

trace_message_level: Current ldap_debug value for the server in hexadecimal form, for example:

- 0x0=0
- 0xffff=65535

trace_message_log: Current LDAP_DEBUG_FILE environment variable setting for the server

The following attributes are used for denial of service prevention:

available_workers: Number of worker threads available for work

current_workqueue_size: Current depth of the work queue

largest_workqueue_size: Largest size that the work queue has reached

idle_connections_closed: Number of idle connections closed by the Automatic Connection Cleaner

auto_connection_cleaner_run: Number of times that the Automatic Connection Cleaner has run

emergency_thread_running: Indicator of whether the emergency thread is running

totaltimes_emergency_thread_run: Number of times the emergency thread has been activated

lasttime_emergency_thread_run: Last time that the emergency thread was activated

The following attribute has been added for alias dereference processing:

bypass_deref_aliases: Server runtime value that indicates whether alias processing can be bypassed (TRUE if no alias object exists in the directory, FALSE if at least one alias object exists in the directory)

The following attributes are used for the attribute cache:

cached_attribute_total_size: Amount of memory used by the directory attribute cache, in KB (includes additional memory used to manage the cache that is not charged to the individual attribute caches, so this total is larger than the sum of the memory that is used by all the individual attribute caches)

cached_attribute_configured_size: Maximum amount of memory, in KB, assigned to the directory attribute cache

cached_attribute_click: Number of times that the attribute has been used in a filter that can be processed by the attribute cache, reported as follows:
cached_attribute_click=attrname:####

cached_attribute_size: Amount of memory that is used for this attribute in the attribute cache, reported in KB as follows:
cached_attribute_size=attrname:####

cached_attribute_candidate_click: A list of up to 10 most frequently used noncached attributes used in a filter that could have been processed by the directory attribute cache if all attributes used in the filter had been cached, reported as follows:
cached_attribute_candidate_click=attrname:####

Analyzing log files

The simplest way to get to a problem is to know the time that it occurred. The log files are time-stamped. Therefore, when you compare the different log files simultaneously for the activities at a certain time, you get very close to knowing the cause of the problem. If multiple LDAP servers are involved (for example, debugging a replication issue), synchronizing their times is helpful (only, of course, if time synchronization is feasible).

Tips:

- ▶ Turning on any 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. For example, the Directory Server might have to write to four places rather than one. Therefore, it is strongly advised that such options be turned on only if the Directory Server is performing poorly and needs to be tuned. Disable these options when you finish your problem analysis.
- ▶ Enabling the change log is seen as a performance bottleneck because the Directory Server must write to the LDAP database and log the relevant information in the change log database. Therefore, it is advisable to enable the change log only if a problem is being debugged or if another application in your organization (that is, a meta directory tool) requires it to be on.
- ▶ Increasing the amount of diagnostic output can result in both performance degradation and insufficient storage conditions in your database file system. Use this procedure only when troubleshooting problems requires the additional diagnostics.

The following sections explain typical steps for monitoring Directory Server status and troubleshooting any problems.

Web administration tool

In the web administration tool, the log files 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, check the following files for error messages first:

- ▶ `ibmslapd.log`
- ▶ `db2cli.log`

The default location is `/var/ldap` for both Oracle Solaris and IBM AIX operating systems.

You can change the location of both of these files by modifying the **ibm-slapdErrorLog** and **ibm-slapdCLIErrors** parameters in `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.

Use these commands 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 finish. This information can, in turn, lead you to missing indexes or unusual directory topology. To turn on the `ibmslapd` trace, run the following commands:

```
ldtrc on
ibmslapd -h 4096
```

After you turn the trace on, run the commands that you think might be causing trouble. Running a trace on several operations can slow performance, so remember to turn off the trace 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 must enable it by using either the web administration tool or the command line.

To view the audit log by using 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 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 the current performance of the directory is necessary to determine what tuning might be necessary, if any.

The ITDSAUDIT.JAR file (covered in 7.1, “ITDSAUDIT.JAR” on page 44) provides a snapshot of the directory and the operations that are run against it by parsing and reporting the transactions that are recorded by the directory audit log.



G

Additional material

This IBM Redpaper publication lists additional material that was referenced in this paper that that can be downloaded from the Internet.

Where to find the web material

The web material cited in this IBM Redpaper is available with the softcopy, online:

<ftp://www.redbooks.ibm.com/redbooks/REDP4258>

Or you can go to the IBM Redbooks web page, select **Additional materials** and open the directory that corresponds with this IBM Redpaper form number, REDP4258:

ibm.com/redbooks

How to use the web material

The additional web material that accompanies this IBM Redpaper is in this file:

REDP4258.zip

This compressed file contains numerous scripts, LDAP Data Interchange Format (LDIF) files, spreadsheets, and other documents that are used and referenced throughout this IBM Redpaper.

Create a subdirectory (folder) on your workstation, and decompress the contents of the .zip file in that 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 publication. Some of the documents referenced here might be available in softcopy only.

IBM Redbooks

For information about ordering this book, see “How to get IBM Redbooks publications” on page 242.

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

Other publications

These publications are also relevant for further information:

- *IBM Security Directory Server Administration Guide Version 6.3.1*, SC27-2749-01
- *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 (IBM Press, July 2003)

Online resources

These websites are also relevant as further information sources:

- IBM Security Systems Information Centers for Security Directory Server 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 Version 8 Product Manuals:
<http://www.ibm.com/software/data/db2/udb/support/manualsv8.html>
- IBM DB2 Information Center:
<http://publib.boulder.ibm.com/infocenter/db2help/index.jsp>
- DB2 articles and resources, IBM developerWorks, Information Management section:
<http://www.ibm.com/developerworks/db2/>

- ▶ DB2 library:
<http://www.ibm.com/software/data/db2/library>
- ▶ IBM Support, 1-800-IBM-SERV:
<http://www.ibm.com/software/support/probsub.html>
- ▶ DB2 UDB Version 8 fix packs and clients:
<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/rpmggroups.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 publications

You can search for, view, or download IBM Redbooks, Redpapers, Hints and Tips, draft publications and additional materials or order hardcopy IBM Redbooks or CD-ROMs at this website:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services



Redpaper

Performance Tuning for IBM Security Directory Server

**Directory Server
performance tuning
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, managing a directory server with several million external users in all 24 time zones throughout the world is a much more daunting task.

IBM Security Directory Server software can handle millions of entries, given the right architecture, configuration, and performance tuning. However, that tuning can differ greatly from tuning for a smaller server with only a few hundred thousand entries. Managing and tuning a directory server of this size requires a change in mindset. Tuning and performance must be a focus even before the hardware is ordered. A proactive approach must be taken after installation also, including the pretuning steps to better interface with other products to make installations and migrations successful, and then regular maintenance to keep the directory running smoothly.

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

In large directory server deployments, use this document as a guide for 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