IBM Security Access Manager

Version 9.0

Performance Tuning Guide

Note
Before using this information and the product it supports, read the information in "Notices"

Edition notice

Note: This edition applies to version 8, release 0, modification 4 of IBM Security Access Manager (product number 5724-C87) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

About this publication

IBM Security Access Manager is a user authentication, authorization, and web single sign-on solution for enforcing security policies over a wide range of web and application resources.

The *IBM Security Access Manager: Performance Tuning Guide* provides conceptual, procedural, and reference information for the performance tuning operations of Security Access Manager.

Intended audience

This guide is for system administrators responsible for determining the resource requirements for and tuning of the Security Access Manager and supported directory servers.

Readers must be familiar with the following topics:

- Microsoft Windows, Linux, and UNIX operating systems
- Database architecture and concepts
- Security management
- Internet protocols, including HTTP, HTTPS, TCP/IP, File Transfer Protocol (FTP), and Telnet
- Lightweight Directory Access Protocol (LDAP) and directory services
- A supported user registry
- Authentication and authorization
- Security Access Manager security model and its capabilities

If you are enabling Secure Sockets Layer (SSL) communication, ensure that you are familiar with SSL protocol, key exchange (public and private), digital signatures, cryptographic algorithms, and certificate authorities.

Chapter 1. Resource usage

This chapter provides information about disk and memory requirements, and CPU and disk speed considerations for Tivoli Directory Server and its prerequisite product, IBM DB2, as used specifically by Security Access Manager.

Resource usage overview

Security Access Manager uses the directory server to store configuration and user information. This information is stored in LDAP entries. User information consists of the user LDAP entry and one or more associated Security Access Manager LDAP entries. As the number of users grows, the disk and memory requirements become primarily based upon the number of Security Access Manager users that exist or that are to be loaded.

The two main situations to consider for disk and memory usage are:

- Directory server run time operation, including search and update used by Security Access Manager to perform authentication and administration.
- Using the directory server **bulkload** utility to load Security Access Manager users.

Tivoli Directory Server **bulkload** utility resource requirements are typically greater than run time operation requirements. The run time operation of the directory server has one requirement, transaction log space, that the bulkload utility does not.

When an environment consisting of multiple replicated directory servers is to be set up, the **bulkload** utility can be used on a single machine to create an initial directory server. The database can then be backed up and restored on all other servers in the environment. With this approach, the **idsdbback** and **idsdbrestore** commands can be used to back up and restore both the database and the directory server configuration.

Another approach to setting up an environment consisting of multiple replicated directory servers is to simultaneously load all the replicated servers using the **bulkload** utility.

For information about the **bulkload** utility, refer to section on “Tivoli Directory Server bulkload utility”.

For information about backing up the database, refer to section on “Backing up the Tivoli Directory Server”.

Disk requirements

Disk space requirements overview

Disk space requirements vary with number of users in the directory, and for the **bulkload** utility, the number of LDAP entries per bulkload pass (**-k** option) and the size of the largest group that is loaded.

The disk requirements discussed in this section consider the number of users in the directory, and are in addition to the disk requirements for a freshly configured empty directory server. The Tivoli Directory Server, through its use of DB2, requires disk space for DB2 transaction logs and for four DB2 table spaces. The default location of these files is under the same directory as the DB2 database that is specified at Tivoli Directory Server configuration time.

For information about how to change the location of these database files, see sections:

- Tuning the DB2 parameters for the transaction logs.
- Expanding storage area options for the table spaces.

The following table summarizes the per user disk space requirements for the run time operation of the directory server. These include the space for holding the users in the DB2 database as well as the transaction log space.

Description	Disk Space Requirement
Transaction logs	410 bytes per user (1)
Tablespace 1: TEMPSPACE1	0 (2)
Tablespace 2: USERSPACE1	3.6KB to 5.4KB per Security Access Manager user (3,4)
Tablespace 3: LDAPSPACE	1.5KB to 2.6KB per Security Access Manager user (3, 4)

1. For the worst case of ACL propagation.
2. Can grow if complex searches are used.
3. Depends on minimal (new) versus standard (old) Security Access Manager user data format. The larger number is for standard format. The minimal format is introduced in IBM Tivoli Access Manager for e-business, version 6, but only available if there are no Security Access Manager users in the directory server before configuration or the users are migrated to the minimal data format using the **amldif2v6** utility.
4. The disk space requirements for a backup of the directory server DB2 database using the DB2 backup command are equal to the combined disk space requirements for tablespace 2 and tablespace 3MB plus 300MB.

The following table summarizes the per user disk space requirements for the **bulkload** utility used to load Security Access Manager users into the directory server. This includes the disk space requirements unique to the bulkload utility as well as the disk space requirements for holding the users in the directory server DB2 database.

Description	Disk Space Requirement
Tablespace 1: TEMPSPACE1	2GB
Tablespace 2: USERSPACE1	3.6KB to 5.4KB per Security Access Manager user (1)
Tablespace 3: LDAPSPACE	1.5KB to 2.6KB per Security Access Manager user (1)
Bulkload input LDIF file	0.8KB to 1.2KB per Security Access Manager user (1)
Bulkload temporary import files for Security Access Manager users	2.4KB * <number of LDAP entries per pass>/2 (2) 3.8KB * <number of LDAP entries per pass>/4 (3)
Bulkload temporary import files for LDAP group	130 bytes * <number f members of the largest group>

1. Depends on minimal (new) versus standard (old) Security Access Manager user data format. The larger number is for standard format. The minimal format was introduced in IBM Tivoli Access Manager for e-business, version 6, but only available if there are no Security Access Manager users in the directory server before configuration or the users are migrated to the minimal data format using the **amldif2v6** utility.
2. For minimal (new) Security Access Manager user data format.
3. For standard (old) Security Access Manager user data format.
4. Not needed for Tivoli Directory Server version 6.0 bulkload.

The following sections provide detail on the disk space requirements for the database files.

Transaction logs

The Tivoli Directory Server uses transaction log disk space for storing uncommitted DB2 transactions from directory update operations.

The transaction log size is limited both by DB2 parameter values and the available disk space. If the transaction logs exceeds the limit due to the settings of the DB2 size parameter, the transaction is backed out using the information in the transaction logs and the transaction fails. If the transaction logs exceed the limit due to a lack of available disk space, the database becomes corrupted and goes into an unusable state.

If the database becomes corrupted in this way, it is possible, but more complex, to issue DB2 commands to recover the database. Alternatively, the database can be restored from a backup or reloaded.

In general, regular backups help reduce the complexity of recovering from failures. The best practice is to ensure there is enough disk space to allow the transaction logs to grow to the DB2 parameter limits.

Note: If the database becomes corrupted, often the recovery commands can be found in the sqllib/db2dump/db2diag.log file which is located in the DB2 instance owner's home directory.

In normal run time use of the Tivoli Directory Server, the transaction log requirements are small. For the **bulkload** utility, the transaction log requirements are even smaller. Some run time directory operations increase the transaction log requirements for a short period.

Here are a few examples:

- The **ldapadd** or **ldapmodify** commands use an increasing amount of transaction log space as the number of multi-valued attributes added to a single LDAP entry in a single command grows, for example when loading many members into a group.

  The transaction log requirements to load a 3M user group are around 300MB.

  Note: The Tivoli Directory Server **bulkload** command with the (**-G**) option can be used as an alternative to using the **ldapadd** and **ldapmodify** commands to add many members to an existing group without using transaction log space.
- An ACL placed on a suffix object can result in that ACL propagating to every entry under that suffix.

  The Tivoli Directory Server performs ACL propagation as one single committed DB2 transaction.

  The transaction log requirements to propagate an ACL to a suffix with 3M Security Access Manager users in the standard (old) data format is around 1.2GB.

  The Security Access Manager fixacls.sh tuning guide script has the same transaction log requirement as ACL propagation, since it performs a similar function.

You can verify that there is sufficient file space for transaction log growth and increase the transaction log space. See "Tuning the DB2 transaction log size".

Table spaces

The Tivoli Directory Server uses four DB2 table spaces.

Tablespace 0: SYSCATSPACE

SYSCATSPACE is used to store a description of the database and its structure and contents. The disk requirements for this table space do not change with the size of the directory. Its disk space requirements are covered by the default directory server disk requirements.

Tablespace 1: TEMPSPACE1

TEMPSPACE1 holds temporary data for sorting and collating DB2 results. The disk requirements for this table space do not change with Security Access Manager's run time use of the directory server. The disk requirements for this table space grow at run time if a complex search is performed on the directory server. The disk space requirements for this table space also grow with the usage of the **bulkload** utility.

The **bulkload** utility disk requirements for this table space are a maximum of approximately 2 GB. The **bulkload** utility uses this maximum space when millions of users are loaded into the directory server.

Tablespace 2: USERSPACE1

USERSPACE1 holds the portion of the database that contains the attribute tables and attribute table indexes for the directory server. These tables are used for optimizing searches on specific attributes.

The disk requirements for USERSPACE1 are approximately 5.4KB per users in the standard (old) Security Access Manager data format (before IBM Tivoli Access Manager for e-business, version 6.0) and 3.6KB per users in the minimal (new) Security Access Manager data format (introduced in IBM Tivoli Access Manager for e-business, version 6.0).

The disk requirements grow from this minimum as the number of optional attributes attached to a user LDAP entry is increased.

Tablespace 3: LDAPSPACE

LDAPSPACE holds the portion of the database that contains the LDAP entry table and LDAP entry table indexes for the directory server. The LDAP entry table contains a few searchable attributes, such as Distinguished Name (DN), and a full, non-searchable definition of each LDAP entry.

The LDAP entry table is used to return the requested attributes from an LDAP search once the search has been narrowed down to a specific entry or set of entries.

The disk requirements for LDAPSPACE are approximately 2.6 KB per users in the standard (old) Security Access Manager data format (before IBM Tivoli Access Manager for e-business, version 6.0) and 1.5KB per users in the minimal (new) Security Access Manager data format (introduced in IBM Tivoli Access Manager for e-business, version 6.0).

Combining Tablespace 2 and 3

Combining the disk requirements for tablespace 2 (USERSPACE) and tablespace 3 (LDAPSPACE), the requirements are approximately 8.0KB per users in the standard (old) Security Access Manager data format (before IBM Tivoli Access Manager for e-business, version 6.0) and 5.1KB per users in the minimal (new) Security Access Manager data format (introduced in IBM Tivoli Access Manager for e-business, version 6.0).

For example, a 1M Security Access Manager user directory server requires about 8GB of disk space in the standard (old) Security Access Manager data format (before IBM Tivoli Access Manager for e-business, version 6.0) and 5.1GB of disk space in the minimal (new) Security Access Manager data format (introduced in IBM Tivoli Access Manager for e-business, version 6.0).

Bulkload disk requirements

In addition to the run time disk requirements, the bulkload utility requires disk space for the input LDIF file, the bulkload temporary import files, and the temporary files for the Security Access Manager tuning guide script, if used.

These requirements are as follows:

Input LDIF file

The input LDIF file requires approximately 1.2KB per user in the standard (old) Security Access Manager data format and 0.8 KB per user in the minimal (new) Security Access Manager data format.

Bulkload temporary import files for Security Access Manager users

The bulkload temporary import files are used to hold the DB2 load tables for loading the database. The bulkload temporary import files for loading Security Access Manager users require approximately 3.8KB per user in the standard (old) Security Access Manager data format and 2.4KB per user in the minimal (new) Security Access Manager data format.

The disk requirement varies with the number of LDAP entries given to bulkload per pass. With Tivoli Directory Server 6.0, these are controlled using the **-k** option.

The per pass disk requirement for the standard Security Access Manager data format is calculated as:

3.8KB * <*number of LDAP entries per pass*>/4

The per pass disk requirement for the minimal Security Access Manager data format is calculated as:

2.4KB * <number of LDAP entries per pass>/2

For example, with a <*number of LDAP entries per pass*> (**-k** option) setting of 4000000 (four million), the bulkload temporary import file disk space requirements are 3.6GB for the standard Security Access Manager data format and 4.6GB for the minimal Security Access Manager data format.

The disk requirements for the minimal data format are higher than for the standard data format, since the minimal data format is denser in the number of attributes per LDAP entry.

Note: Given the same number of LDAP entries per pass, the minimal data format requires only half the number of passes as the standard data format requires. This is due to a reduction in the number of LDAP entries per user in the minimal data format. In the minimal format there are only two entries per user and in the standard, there are four entries per user.

Bulkload temporary import files for groups

The bulkload temporary import files for loading an LDAP group require approximately 130 bytes per member added to the group.

Note that with Tivoli Directory Server version 6, a large group does not have to be loaded in a single usage of **bulkload** command. The group member list can be divided into multiple member lists. Each member list is added to the group using separate bulkload commands with **-G** option.

Memory requirements

The run time memory requirements for the Tivoli Directory Serverand prerequisite product, IBM DB2, vary with number of users in the directory server, the size of the directory server caches, and the size of the DB2 buffer pools.

As a rough estimate, a directory server with less than a million users requires a system with 2 GB of memory. A directory server with more than a million users requires a minimum of 2GB, but performs better with 4 GB of memory. For directory servers with several million users or more, the most beneficial allocation of excess memory for performance is to the DB2 buffer pools.

For more information about tuning the DB2 buffer pools, see the "Tuning the DB2 buffer pools" section.

You can improve bulkload performance and remove a redundant level of caching by disabling file system caching.

CPU and disk speed guidelines

CPU and disk-speed requirements are directly related to throughput requirements and the number of users in the directory server.

For directory sizes of less than a million users, directory server caches and DB2 buffer pools can be made large enough to cache most of the database and CPU becomes the bottleneck to performance.

For directory sizes greater than a million users, caching tends to be less possible and the disk system becomes the bottleneck.

Guidelines for CPU and disk-speed requirements:

- For search performance, such as Security Access Manager authentication, each added replica servers increases the maximum throughput rate linearly.

- For update performance, such as creating new Security Access Manager users, each added replica server has no impact on the throughput rate.

  If there are millions of users, the Directory Proxy Server can be used to improve update performance. Each added back-end directory server in a proxy environment increases the maximum update rate linearly.
- Update performance is typically bottle-necked by disk system performance. A faster disk system improves update rates.
- If the CPU and disk speed of a replica server is not equal to or greater than the master server, a sustained rate of updates results in the replica server getting behind on updates.
- For systems bottle-necked on disk I/O due to high update rates or due to having too many users to benefit from LDAP or buffer pool caching, spreading the database across multiple disk drives results in a large improvement to performance.

  The change to performance or magnitude of the improvement decreases with each added disk drive.

Chapter 2. Tuning Tivoli Directory Server

This chapter provides sequential and detailed information about tuning the Tivoli Directory Server.

Security Access Manager 9.0 configurations do not support the tuning guide scripts mentioned in this document. Many of the scripts are only relevant for the previous software stack versions of Security Access Manager.

You must perform tuning manually, as described in this chapter.

Tivoli Directory Server tuning overview

This section provides information about tuning the Tivoli Directory Server for best performance with Security Access Manager.

To make tuning easier, see following information:

- Tuning the Tivoli Directory Server for general use.
- Configuration-related tasks.

Configuration tasks are included because certain parts of the directory server must be configured before they are tuned.

Before you use the information to tune the Tivoli Directory Server, the server must be configured with at least one instance and database. If the directory server is loaded with users and data, it is a good idea to back up the directory server in case failures occur during the tuning process.

For more information about the Tivoli Directory Server backup options, "Backing up the Tivoli Directory Server".

For more information about configuring the directory server, see the Tivoli Directory Server documentation.

For information about installing and configuring Tivoli Directory Server, see the *IBM Security Access Manager: Installation Guide* or the Tivoli Directory Server documentation.

A basic understanding of LDAP, and of the design and deployment of a user namespace on LDAP, are some of the prerequisites to understand the procedures.

For more information, see the following IBM Redbook:

- *Understanding LDAP*

See the following topic from the IBM Systems Journal:

- An Enterprise Directory Solution with DB2 http://www.research.ibm.com/journal/sj/392/shi.pdf

When to tune Tivoli Directory Server

This section provides tips on the optimal times for tuning Tivoli Directory Server.

The best time to tune the Tivoli Directory Server is immediately after it has been configured. If the Tivoli Directory Server has never been tuned, it performs poorly as the size of the directory grows, for example as users are added to the server.

Poor performance includes slow search and update response time and slow execution time for the **bulkload** utility. Tune the Tivoli Directory Server at least once, and then update the DB2 system statistics after any large number of updates to the server. For example after adding many users.

The DB2 system statistics provide information to the DB2 optimizer so it can decide how to optimize query operations. Updating the DB2 system statistics involves running DB2 **runstats** command or the **reorgchk** command, setting all tables to volatile and overriding system statistics for Tivoli Directory Server and Security Access Manager.

For more information about updating the DB2 system statistics, refer to "Updating the DB2 system statistics"

The fastest way to load and tune the Tivoli Directory Server with millions of users it to configure the directory server, tune the directory server, use the **bulkload** utility to load users into the directory server and finally update the DB2 system statistics. Before tuning the directory server, back it up in case of failures during the tuning procedure.

Refer to "Backing up the Tivoli Directory Server"_for more information about backing up the DB2 database and directory configuration.

For information about each step required to tune the Tivoli Directory Server, refer to "Tivoli Directory Server tuning task roadmap".

For information about the Tivoli Directory Server and tuning for special cases, refer to Chapter 3, "Additional Tivoli Directory Server tuning".

Note regarding restoring the directory:

Unless the optimizations on a backed up DB2 database are known, the Tivoli Directory Server must be retuned after restoring from backup. When a DB2 backup is performed the optimizations on the database are preserved in the backed up database.

For example, the DB2 system statistics and DB2 parameter settings are preserved in the backed up database. When the Tivoli Directory Server **idsdbback** command is used to back up a directory, the directory server configuration is backed up along with the DB2 database.

Tivoli Directory Server tuning task roadmap

This section outlines the steps for configuring and tuning Tivoli Directory Server.

About this task

This a summary of the full set of configuration and performance tuning steps for the Tivoli Directory Server.

Procedure

1. Optional: Back up the Tivoli Directory Server using **idsdbback** or **db2 backup**.

   It is always a best practice to back up the Tivoli Directory Server before you make any major change.

   See "Backing up the Tivoli Directory Server".
2. Optional: Expand your storage capabilities. See "Expanding storage area options".
3. If the Tivoli Directory Server has never been started, start it now to complete the Tivoli Directory Server configuration.

   DB2 database tables are not created until the first startup of the Tivoli Directory Server.

   See "Starting the Tivoli Directory Server".
4. Tune the AIX, Linux, or Solaris operating system resource limits.

   See "AIX, Linux, or Solaris operating system resource limits."
5. Tune the DB2 database parameters.

   See "Tuning the DB2 database parameters".
6. Create the LDAP_DESC(AEID, DEID) index.

   See "Creating the LDAP_DESC(AEID,DEID) index".
7. Tune the Tivoli Directory Server change log.

   Performance is faster when the change log is not configured.

   See "Tuning the Tivoli Directory Server change log".
8. Tune the Tivoli Directory Server audit-log.

   Performance is faster when the audit-log is turned off.

   See "Tuning the Tivoli Directory Server audit log".
9. Configure the Tivoli Directory Server suffixes.

   See "Configuring the Tivoli Directory Server suffixes".
10. Create LDAP entries for the configured suffixes.

    See "Creating LDAP entries for configured suffixes".
11. Prepare the Tivoli Directory Server for loading Security Access Manager users.

Either configure Security Access Manager into the directory server or do the following steps:

a) Apply the Security Access Manager schema to the Tivoli Directory Server.

b) Create an unconfigured Security Access Manager directory tree.

c) If applicable, create one or more unconfigured Security Access Manager subdomain trees.

See "Loading Security Access Manager users".

12. Tune the Tivoli Directory Server access control lists (ACLs). See Tuning the Tivoli Directory Server ACLs".

13. Tune the DB2 indexes.

    See "Tuning the DB2 indexes".

14. Update the DB2 system statistics.

    Repeat this procedure each time many LDAP entries are added to the directory, for example after loading many users.

    The following steps update the DB2 system statistics:

    a) Update system statistics using DB2 **runstats** or **reorgchk**.

    b) Define DB2 tables as volatile.

    c) Override system statistics for Tivoli Directory Server and Security Access Manager.

15. Optionally, back up the tuned Tivoli Directory Server using **idsdbback** or **db2 backup**.

    See "Backing up the Tivoli Directory Server".

16. Start Tivoli Directory Server.

    See "Starting Tivoli Directory Server".

17. Test the Tivoli Directory Server performance.

    See "Testing the Tivoli Directory Server performance".

DB2 command examples

This section describes example DB2 commands used to tune the DB2 parameters.

If the example commands are to be executed manually, the UNIX login context must be the DB2 database instance owner. The login context can be established as the DB2 instance owner by either logging as the DB2 database instance owner, or by switching to the instance owners login context.

The following are examples of switching to the login context of a DB2 instance owner named **ldapdb2**:

On AIX, Linux, or Solaris operating system, enter the following: su – ldapdb2

On Windows systems, enter the following commands:

```
db2cmd
set DB2INSTANCE=ldapdb2
```

All the DB2 command examples assume the following configuration parameters:

- DB2 instance owner is **ldapdb2**
- DB2 database name is **ldapdb2**

Stopping the Tivoli Directory Server

This section explains how to stop Tivoli Directory Server.

To determine whether the Tivoli Directory Server is running, perform one of the following tasks:

- On UNIX systems, enter:
  ```
  ps -ef | grep slapd
  ```
  If the output shows a process named **slapd** or **ibmslapd**, the Tivoli Directory Server is running.
- On Windows systems, view the status of the Tivoli Directory Server service.

To stop the Tivoli Directory Server server, perform the following tasks:

- On AIX, Linux, or Solaris operating system systems with Tivoli Directory Server version 6.0 or later, enter the following command:
  idsslapd -I *instance* -k
- On Windows operating systems, stop the Tivoli Directory Server service.

1. Backing up the Tivoli Directory Server

This section explains how to back up Tivoli Directory Server.

Before performing this task, stop the Tivoli Directory Server. See "Stopping the Tivoli Directory Server".

There are several ways to back up and restore the Tivoli Directory Server each with its own set of advantages and disadvantages. This section describes some of them.

idsdbback and idsdbrestore commands

This section explains how to use the **idsdbback** and **idsdbrestore** to back up Tivoli Directory Server.

The **idsdbback** and **idsdbrestore** commands are provided by Tivoli Directory Server to back up both the DB2 database and the directory server configuration. The advantage to using these commands is the backing up of the directory server configuration.

The other ways of backing up the directory server do not include the directory server configuration information. This information includes directory schema, configuration file and key stash file. Although it is possible to back up these files manually, the **idsdbback** and **idsdbrestore** commands make this task easier.

The disadvantage to using the **idsdbback** and **idsdbrestore** commands is less flexibility in how the underlying DB2 restore is performed. For example, using the **idsdbrestore** command, the DB2 restore cannot be directed to distribute the database across multiple disks. Another disadvantage is the potential incompatibility in backing up the database on one platform (for example AIX), and restoring on another platform (for example, Windows).

db2ldif, ldif2db, and bulkload commands

Use the **db2ldif** and **ldif2db** commands to dump data to a database or to restore a database from a file in Lightweight Directory Interchange Format (LDIF). The advantage to using these commands is portability, standardization, and size.

The output LDIF file can be used to restore a directory server on a different platform and possibly with some modification on a different directory server product, for example the Sun Java System Directory. Because the LDIF format is text-based, an LDIF file is relatively easy to modify for any differences between LDAP products.

The disk space for an LDIF file containing 2M Security Access Manager users and a single static group containing all users in the minimal data format is 1.53 GB. The disk space for a DB2 backup of the same users is 9.94 GB. The disk space requirement for the LDIF output from **db2ldif** is approximately 6.5 times less than the backed up database from **db2 backup** or **idsdbback.**

The disadvantage to using **db2ldif** and **ldif2db** is speed as compared to **db2 backup** and **db2 restore**. On a system with a single disk drive and a database with 2M Security Access Manager users in the minimal data format, the **db2ldif** command takes about 3 hours to dump the database in LDIF format.

For restoring the database, the **bulkload** utility is many times faster than **ldif2db** command, yet is still much slower than **db2 restore**. For the same 2M Security Access Manager users in the minimal data format, the **bulkload** utility takes 4 hours to load the LDIF file back into the directory server. In the same environment, the **db2 backup** and **db2 restore** commands each take 20 minutes apiece to complete. The **bulkload** utility takes approximately 12 times longer than a **db2 restore**. Because the **db2 backup** and **restore** commands are essentially a disk copy, they are the fastest alternative.

Unlike **db2 restore**, in order to restore a directory server using **bulkload**, the directory server must be empty. This is accomplished by restoring an empty database, or by unconfiguring, then reconfiguring the directory server. Do not drop the database and recreate it using DB2 commands. Manually creating the database misses some important configuration steps that are part of the Tivoli Directory Server configuration tool.

An important consideration when using **db2ldif** is a previous use of the fixacls.sh script or one of its variations from a previous release. After using the fixacls.sh script, the output of the **db2ldif** command does not contain the correct access control lists (ACLs) on Security Access Manager entries.

To ensure that the correct Security Access Manager ACLs are defined after restoring the directory server, for example using **bulkload**, from the output of **db2ldif**, rerun the fixacls.sh script after the restore.

For more information, see “Using the fixacls.sh scipt”.

DB2 backup and restore commands

The **db2 backup** and **db2 restore** commands are provided by IBMDB2. The advantage to using these commands is performance and flexibility for specifying the location of the database files.

The **db2 restore** command can be used to distribute the database across multiple disks or to simply move the database to another directory. See “Distributing the database across multiple physical disks” for more information.

The disadvantage to the **db2 backup** and **db2 restore** commands is their complexity. Some of the complexity of the DB2 commands can be overcome by using the am_tune_ldap.sh tuning script. For example, the am_tune_ldap.sh tuning script ensures that the permissions on the backup directory allow the backup to succeed.

Another disadvantage is the potential for incompatibility in backing up and restoring across platforms and across DB2 versions. For more information, see the appropriate DB2 backup and restore documentation.

An important consideration when using **db2 backup** and **db2 restore** commands is the preservation of DB2 configuration parameters and system statistics optimizations in the backed-up database. The restored database has the same performance optimizations as the backed-up database. This is not the case with LDAP **db2ldif**, **ldif2db**, or **bulkload**.

If you restore over an existing database, any performance optimization on that existing database are lost. Check all DB2 configuration parameters after performing a restore. Also, if you do not know whether a **db2 runstats** was performed before the database was backed up, tune the DB2 system statistics after the restore (see “Updating the DB2 system statistics”). The DB2 commands to perform backup and restore operations are as follows:

```
db2 force applications all
db2 backup db ldapdb2 to directory_or_device
db2 restore db ldapdb2 from directory_or_device replace existing
```

where *directory_or_device* is the name of a directory or device where the backup is stored.

The most common error that occurs on a restore is a file permission error. Following are some reasons why this error might occur:

- 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:

- The backed-up database is distributed across multiple directories, and those directories do not exist on the target system of the restore.

  Distributing the database across multiple directories is accomplished with a redirected restore. To solve this problem, either create the same directories on the target system or perform a redirected restore to specify the proper directories on the new system. If creating the same directories, ensure that the owner of the directories is the DB2 instance owner typically the ldapdb2 user.

  For more information about redirected restore, see “Distributing the database across multiple disks”.

2. Expanding storage area options

When the database reaches its disk limit, expand the data storage.

You can expand the data storage by:

- Using Storage Area Networks, which is not discussed in detail in this guide.
- Distributing the database across multiple physical disks. See “Distributing the database across multiple physical disks”.
- Use raw devices. See “Using the raw device support for DMS table spaces”.

Distributing the database across multiple physical disks

Distribute the database across multiple physical disks to improve performance.

Distributing the database across multiple disks also improves performance. One 20 GB disk is slower than two 10 GB disks. This is due to the concurrency that can be achieved with multiple disk drives.

Before you do this task, stop the Tivoli Directory Server. See “Stopping the Tivoli Directory Server”.

Setting up the file system permission and using the DB2 commands to distribute the database across multiple disks can be complex tasks. The am_tune_ldap.sh tuning script reduces some of the complexity for Linux and UNIX operating systems. The am_tune_ldap.sh script verifies the permissions on the file systems and in some cases corrects them. The am_tune_ldap.sh script issues the DB2 backup and restore and related commands to distribute the database.

Before you use the am_tune_ldap.sh script to distribute the database, edit it and search for the variables named tablespace2containers and tablespace3containers. These variables specify the file system locations, called container paths, that are used for distributing the database. The script contains several example usages of these variables.

Viewing Tivoli Directory Server table spaces

Tivoli Directory Server creates the directory database with the **db2 create database** command from the database name that is specified with the directory configuration command.

Before you begin

Switch context to the DB2 instance owner.

About this task

Tivoli Directory Server creates the directory database with four System Managed Space (SMS) table spaces.

Procedure

1. Run one of the following commands:

   Note: These steps use ldapdb2 as the DB2 instance owner. Change it as appropriate for your environment.

   On AIX, Linux, and Solaris operating systems:

   ```
   su – ldapdb2
   ```

   On Windows operating systems:

   ```
   db2cmd
   set DB2INSTANCE=ldapdb2
   ```

2. View the table spaces by running the following DB2 commands under the context of the DB2 instance owner:

   ```
   Tablespaces for Current Database

   Tablespace ID= 0
   Name= SYSCATSPACE
   Type= System managed space
   Contents= Any data
   State= 0x0000 Detailed explanation:
   Normal
   ```

Example:

The following examples show UNIX table space output for Tivoli Directory Server:

```
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= LDAPSPACE1
Type= System managed space
Contents= Any data
State= 0x0000
Detailed explanation:
Normal
```

The Tivoli Directory Server is stored in the user table space (USERSPACE1) and in the LDAP table space (LDAPSPACE). By default, there is only one container or directory for each of these table spaces. To view the details about the user table space, enter a DB2 command like this one:

```
db2 list tablespace containers for 2
```

Example output:

```
Tablespace Containers for Tablespace 2

Container ID= 0
Name= /ldapdb2/NODE0000/SQL00001/SQLT0002.0
Type= Path
```

The container or directory that DB2 uses for table space 2 is /ldapdb2/SQL00001/SQLT0002.0. It contains LDAP attribute database tables.

Table space 3 contains the ldap_entry table.

For more information about table spaces, see “Table spaces”.

Creating file systems and directories on the target disks

This section explains the steps for distributing the DB2 database by creating file systems and directories across multiple 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 across which the database will be distributed.

Guidelines for this task include:

- Because DB2 distributes the database equally across all directories, make all the file systems, 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. Instead of deleting the lost+found directory, create a subdirectory at the root of each file system to be used for distributing the database. For example, create a subdirectory named ldapdb2_containers in each file system where the DB2 database is to be stored.
- Create two additional directories under the ldapdb2_containers directory: one for holding tablespace 2 and the other for tablespace 3.

  For example, these directories might be named tblspc2 and tblspc3. Then specify these directories on the **set tablespace** commands as discussed in "Performing a redirected restore of the database".
- The DB2 instance user must have Write permission on the created directories.

  For AIX and Solaris systems, the following command gives the proper permissions: chown ldapdb2 directory_name

  Platform-specific guidelines include:
- For the AIX operating system, create an Enhanced Journaled Files System (JFS2) or create the file system with the Large File Enabled option.

  This option is one of the options on the Add a Journaled File System **smit** menu.
- For AIX and Solaris systems, set the file size limit to unlimited or to the size of the largest file system under which the DB2 database files reside.

  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

This section explains how to back up the database.

About this task

To back up the existing database, follow these steps:

Procedure

1. Stop the Tivoli Directory Server server process (**slapd** or **ibmslapd**).
2. To close all DB2 connections, enter:
   ```
   db2 force applications all
   db2 list applications
   ```
   A message like this one is displayed:
   ```
   SQL1611W No data was returned by Database System Monitor.
   ```
3. To initiate the backup process, enter:
   ```
   db2 backup db ldapdb2 to [file system | tape device]
   ```
   When the database has been backed up successfully, a message like this is displayed:
   ```
   Backup successful. The timestamp for this backup image is : 20000420204056
   ```

Results

The database is backed up.

Note: Ensure that the backup process was successful before proceeding. The next step destroys the existing database in order to recreate it. If the backup was not successful, the existing database is lost. You can verify the success of the backup by restoring to a separate system.

Performing a redirected restore of the database

A DB2 redirected restore restores the specified database table space to multiple containers or directories. This section explains how to perform a redirected restore of the database.

About this task

In the following example, assume that the following directories for containing tablespace 2 were created, are empty, and have the correct permissions to allow write access by the DB2 instance owner, typically the **ldapdb2** user:

```
/disk1/ldapdb2_containers/tblspc2
/disk2/ldapdb2_containers/tblspc2
/disk3/ldapdb2_containers/tblspc2
/disk4/ldapdb2_containers/tblspc2
/disk5/ldapdb2_containers/tblspc2
```

In the following example, assume the following directories for tablespace 3 were created:

```
/disk1/ldapdb2_containers/tblspc3
/disk2/ldapdb2_containers/tblspc3
/disk3/ldapdb2_containers/tblspc3
/disk4/ldapdb2_containers/tblspc3
/disk5/ldapdb2_containers/tblspc3
```

Follow these steps for a redirected restore:

Procedure

1. To start the DB2 restore process, enter:

   ```
   db2 restore db ldapdb2 from [location of backup] replace existing redirect
   ```

   Messages like these ones are displayed:

   ```
   SQL2539W Warning! Restoring to an existing database that is the same as the backup image
   database. The database files will be deleted.

   SQL1277N Restore has detected that one or more tablespace containers are inAccessible, or
   has set their state to ’storage must be defined’.

   DB20000I The RESTORE DATABASE command completed successfully.
   ```

   This command prepares for the restore, but does not actually perform the restore. These messages indicate that DB2 is prepared to receive the next commands, which define the location of the database files.

2. To define the containers for tablespace 2 and for tablespace 3, enter:

   ```
   db2 "set tablespace containers for 2 using (path \
   ’/disk1/ldapdb2_containers/tblspc2’, \
   ’/disk2/ldapdb2_containers/tblspc2’, \
   ’/disk3/ldapdb2_containers/tblspc2’, \
   ’/disk4/ldapdb2_containers/tblspc2’, \
   ’/disk5/ldapdb2_containers/tblspc2’ )"

   db2 "set tablespace containers for 3 using (path \
   ’/disk1/ldapdb2_containers/tblspc3’, \
   ’/disk2/ldapdb2_containers/tblspc3’, \
   ’/disk3/ldapdb2_containers/tblspc3’, \
   ’/disk4/ldapdb2_containers/tblspc3’, \
   ’/disk5/ldapdb2_containers/tblspc3’ )"
   ```

   Note: If many containers are defined, these commands can become too long to fit within the limits of a shell command. In this case, you can put the command in a file and run within the current shell using the dot notation. For example, assume that the commands are in a file named set_containers.sh. The following command runs it in the current shell:

   ```
   . set_containers.sh
   ```

   After completion of the DB2 **set tablespace** command, a message like this is displayed:

   ```
   DB20000I The SET TABLESPACE CONTAINERS command completed successfully.
   ```

   If you receive the following message:

   ```
   SQL0298N Bad container path. SQLSTATE=428B2
   ```

   This indicates that one of the containers is not empty, or that Write permission is not enabled for the DB2 instance owner, typically the **ldapdb2** user:

Note: A newly created file system on AIX and Solaris contains a directory named lost+found. You should create a directory at the same level as lost+found to hold the table space and then reissue the **set tablespace** command. If you experience problems, see the DB2 documentation. The following files might also be of interest:

ldapdb2_home_dir /sqllib/Readme/en_US/Release.Notes
ldapdb2_home_dir /sqllib/db2dump/db2diag.log

Note the db2diag.log file contains some fairly low-level details that can be difficult to interpret.

3. Continue the restore to new table space containers. This step takes the most time to complete. The time varies depending on the size of the directory. To continue the restore to the new table space containers, enter:

   ```
   db2 restore db ldapdb2 continue
   ```

   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
   ```

Using raw device support for DMS table spaces

With Tivoli Directory Server, versions 6.2 and 6.3, you can select the type of table space. You can use raw device support for Database Managed Space (DMS) table spaces.

The options are Database Managed Space (DMS) or System Managed Space (SMS).

You can add a raw device and a file to the containers in DB2, versions 9.5 and 9.7. Raw devices provide an alternative way to add multiple physical disks to the containers for LDAP table spaces LDAPSPACE and USERSPACE.

See the Tivoli Directory Server documentation for more information about adding disk space to SMS and DMS table spaces.

3. Starting the Tivoli Directory Server

This section describes how to start Tivoli Directory Server.

After the Tivoli Directory Server server has been configured, it is necessary to complete the database configuration by starting the server.

Database tables and indexes are not defined until the first time the server is started.

To determine whether the Tivoli Directory Server is running, do one of the following:

- On UNIX systems, enter:

  ```
  ps -ef | grep slapd
  ```

If the output shows a process named **slapd** or **ibmslapd**, the Tivoli Directory Server is running.

- On Windows systems, view the status of the Tivoli Directory Server service.

To start the Tivoli Directory Server server, do one of the following:

- On UNIX systems with Tivoli Directory Server version 6.0 or later, enter the following command:
  idsslapd -I *instance*
  where *instance* is the directory server instance name to start.
- On Windows operating systems, start the Tivoli Directory Server service.

4. AIX, Linux, or Solaris operating system resource limits

This section describes changes to the AIX, Linux, or Solaris operating system that are necessary to support directories with millions of users.

Resource limits on AIX, Linux, and Solaris operating systems (ulimit)

This section describes using the **ulimit** command to configure the resource limits on Linux and UNIX.

On AIX, Linux, and Solaris operating systems, the **ulimit** command controls the limits on system resource, such as process data size, process virtual storage, and process file size. Specifically:

- On Solaris operating systems, by default, the **root** user has unlimited access to these resources (for example, unlimited).
- On AIX operating systems, some limits might apply to the **root** user.

On AIX, Linux, or Solaris operating system systems, resource limits are defined for each user. When a process is started, that process inherits or takes on the resource limits of the user context under which it was started.
For example, if the Tivoli Directory Server process, **idsslapd**, is started under the root user context, the **idsslapd** process takes on the resource limits of the root user. This inheritance occurs even if the process switches user contexts as the **idsslapd** process does.

The **idsslapd** process switches the user context to the DB2 instance owner. If the need is to have the **idsslapd** process take on the resource limits of the DB2 instance owner, the **idsslapd** process must be started while the DB2 instance owner is logged in.

Resource limits on Solaris operating system

This section describes the resource limits on Solaris.

Increasing the shared memory maximum (shmmax)

You must increase the shared memory maximum to allow DB2 processes to allocate the buffer pool space. In a later step, you can change the buffer pool settings to sizes that might be too large for the default shared memory maximum.

Set the shared memory size to the size of the physical memory on the machine. For example, if the Tivoli Directory Server machine contains 512 MB of physical memory, set the shared memory maximum size to 540000000.

On Solaris operating systems, update the shared memory maximum in the /etc/system file by changing the following line:

```
set shmsys:shminfo_shmmax = physical_memory
```

where *physical_memory* is the size of the physical memory on the system in bytes. After changing the shared memory maximum, reboot the system for changes to take effect.

Examining the contents of the /etc/system file is not a reliable way to determine the operating system setting for the shared memory maximum. For that purpose, enter the following command:

```
sysdef | grep –i shmmax
```

The following message indicates that the shared memory maximum has not been set large enough for the DB2 cache:

```
SQL1478W The database has been started but only one buffer pool has been activated.
SQLSTATE=01626
```

An insufficient size for the shared memory maximum can also prevent DB2 from starting. In this case, the following message is displayed:

```
SQL1220N The database manager shared memory set cannot be allocated.
```

These messages also appear when starting the Tivoli Directory Server or running the following DB2 command:

```
db2 connect to ldapdb2
```

Setting process limits in Tivoli Directory Server

Tivoli Directory Server, sets the **ulimits** based on the configuration file. You can find this information in the instance configuration as shown in the following example:

```
dn: cn=Ulimits, cn=Configuration
cn: Ulimits
ibm-slapdUlimitDataSegment:262144
ibm-slapdUlimitDescription: Prescribed minimum ulimit option values
ibm-slapdUlimitFileSize:2097152
ibm-slapdUlimitNofile:500
ibm-slapdUlimitStackSize: 10240
ibm-slapdUlimitVirtualMemory:1048576
objectclass: top
objectclass: ibm-slapdConfigUlimit
```

objectclass: ibm-slapdConfigEntry

Resource limits on AIX operating system

When AIX settings take effect

In AIX, system settings take effect when you restart the UNIX shell. Operations (such as su -, login, and a process restart) pick up the new system settings; sudoes do not pick them up. You can temporarily test resource limits by running the **ulimit** command and restarting the affected processes.

Setting process limits in Tivoli Directory Server

Tivoli Directory Server, sets the **ulimits** based on the configuration file. You can find this information in the instance configuration as shown in the following example:

```
dn: cn=Ulimits, cn=Configuration cn: Ulimits
ibm-slapdUlimitDataSegment:262144
ibm-slapdUlimitDescription:Prescribed minimum ulimit option values
ibm-slapdUlimitFileSize:2097152
ibm-slapdUlimitNofile:500
ibm-slapdUlimitStackSize:10240
ibm-slapdUlimitVirtualMemory:1048576
objectclass: top
objectclass: ibm-slapdConfigUlimit objectclass: bim-
slapdConfigEntry
```

Creating file systems with large file support

The standard file system on 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 as Enhanced Journaled File System or JFS2. This option is available starting in AIX version 5.2.

Another 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.

Refer to AIX documentation for additional information and file system options.

Environment variables

About this task

The environment variable SPINLOOPTIME=650 (for SMP systems) improves the performance of the Tivoli Directory Server.

You can set environment variables as shown:

- Temporarily define the environment variables before starting the server, then undefine them. Here is an example:

```
export SPINLOOPTIME=650
ibmslapd
unset SPINLOOPTIME
```

- Pass the environment variable on the server start command. Here is an example:

  SPINLOOPTIME=650 ibmslapd
- For Tivoli Directory Server, define the environment variables in the slapd32.conf or ibmslapd.conf configuration files. For more information, see the Tivoli Directory Server documentation.
- Define the variables in the /etc/environment file.

  For the environment variables in the /etc/environment file to take effect, the user must log off and back on again before starting the server. The advantage to this approach is that it is automatic, thus reducing the chance for error.

  The disadvantage is the potential to affect other processes in the system. At this time, there are no known processes that are detrimentally affected by the setting of these environment variables.

5. Tuning the DB2 database parameters

This section describes how to tune the DB2 database parameters.

Tuning with DB2 versions 9.x

You can use the Self Tuning Memory Manager (a feature of DB2, version 9.0) instead of manually tuning DB2 parameters.

The Self Tuning Memory Manager assigns optimal values to memory consumers based on system usage and available resources. You can enable it by setting:

- SELF_TUNING_MEM to **ON**
- The value of two or more consumers, such as bufferpool or sort memory, to AUTOMATIC.

Use **idsperftune** , the Tivoli Directory Server tool, to facilitate configuration and enable the STMM. You can then enable the following memory consumers for self tuning:

Buffer pools

The ALTER BUFFERPOOL and CREATE BUFFERPOOL statements control the buffer pools.

Package cache

The *pckcachesz* configuration parameter controls the package cache.

Locking memory

The *locklist* and *maxlocks* configuration parameters control the locking memory.

Sort memory

The *sheapthres_shr* and *sortheap* configuration parameters control the sort memory.

Database shared memory

The *database_memory* configuration parameter controls the database shared memory.

Perform the following manual steps to ensure that database memory is properly managed when using the Self Tuning Memory Manager:

1. Ensure the DATABASE_MEMORY setting is on AUTOMATIC (default).
2. If on Solaris or Linux, stop and restart DB2.On these operating systems,DB2 adjusts the size of DATABASE_MEMORY at start-up.
3. Run the system under normal loads to allow DB2 to find the optimal size of DATABASE_MEMORY for the workload.
4. Run the command db2 get db cfg show detail to see the size of DATABASE_MEMORY selected by DB2.
5. Change the DATABASE_MEMORY to the size determined by DB2 instead of AUTOMATIC to optimize performance with a fixed value.
6. (Solaris or Linux only) Do the following tasks:

   Note: On Solaris and Linux, DB2 adjusts the size of DATABASE_MEMORY at start-up. On Windows and AIX, DB2 adjusts the size of DATABASE_MEMORY while the system is under load.

   a) Stop and restart DB2.
   b) Run the system under normal load again.

Before performing these tasks, stop the Tivoli Directory Server. See "Stopping the Tivoli Directory Server".

If the *db2 force applications all* command mentioned in this section is executed manually and the Tivoli Directory Server is running, the server becomes partially functional. Any directory cached searches appear to respond correctly. Other searches might return with no results, or error messages. The recover is to stop and the restart the Tivoli Directory Server.

The Security Access Manager db2_tunings.sh tuning script automates the tasks in this section. The db2_tunings.sh script can be run directly, or as part of the set of tasks performed by the am_tune_ldap.sh tuning script.

All Security Access Manager tuning guide scripts must be started from the context of the UNIX root system user, for example after logging in as the root system user.

Tuning the database configuration parameters

For Tivoli Directory Server 6.2, see "Tuning with DB2 versions 9.x".

Enter the command shown to display the current setting of key DB2 configuration parameters that affect performance:

```
db2 get database configuration for ldapdb2 | egrep 'HEAP|MAXLOCKS|MINCOMMIT'
```

The output is displayed as follows, and shows the default settings after configuring the Tivoli Directory Server:

```
Database heap (4KB)(DBHEAP) = 1200
Utilities heap size (4KB)(UTIL_HEAP_SZ) = 5000
Percent of mem for appl. group heap(GROUPHEAP_RATIO) = 70
Max appl. control heap size (4KB)(APP_CTL_HEAP_SZ) = 128
Sort heap thres for shared sorts (4KB) (SHEAPTHRES_SHR) = (SHEAPTHRES)
Sort list heap (4KB)(SORTHEAP) = 256
SQL statement heap (4KB)(STMTHEAP) = 4096
Default application heap (4KB)(APPLHEAPSZ) = 1280
Statistics heap size (4KB)(STAT_HEAP_SZ) = 4384
Percent. of lock lists per application(MAXLOCKS) = 10
Group commit count(MINCOMMIT) = 1
```

These parameters do not require adjustment from their default settings. Poor performance results from setting MINCOMMIT to anything other than 1. Although not necessary, these parameters can be changed by using a command with the following syntax:

```
db2 update db cfg for ldapdb2 using parm_name  parm_value
db2 terminate
db2 force applications all
```

where *parm_name* is the name of the parameter, shown on the output from the get database configuration command at the left side of the equals sign, and *parm_value* is the new setting.

You can use the DB2 AUTOCONFIGURE command instead of individually setting the parameter, but the default settings are optimized for Security Access Manager in most environments. See DB2 Command Line Processor (CLP) documentation for more information about the AUTOCONFIGURE command.

Incorrect settings for some database parameters can cause database failures. If this is suspected, check the following files for DB2 error messages:

- For Tivoli Directory Server:
  db2instance_owner's_home_directory/idsslapd-*instance name*/logs/db2cli.log
  db2instance_owner's_home_directory/sqllib/db2dump/db2diag.log

Tuning the DB2 buffer pools

DB2 buffer pools are in memory caches that hold database tables and indexes. Buffer pools improve database performance by eliminating expensive disk read time when items are found in the buffer pool.

For Tivoli Directory Server 6.2, see "Tuning with DB2 versions 9.x".

The Tivoli Directory Server has two important buffer pools; one for the table space named USERSPACE1 and one for the table space named LDAPSPACE.The buffer pool for USERSPACE1 is named IBMDEFAULTBP and the buffer pool for the LDAPSPACE table space is named LDAPBP. The following table illustrates this relationship.

Table 1. Tablespace and buffer pool names

Tablespace Number	Tablespace Name	Buffer Pool Name
1	USERSPACE1	IBMDEFAULTBP
2	LDAPSPACE	LDAPBP

USERSPACE1 holds the portion of the database that contains the attribute tables and attribute table indexes for the directory server. These tables are used for optimizing searches on specific attributes.

LDAPSPACE holds the portion of the database that contains the LDAP entry table and LDAP entry table indexes for the directory server. The LDAP entry table contains a few searchable attributes, such as Distinguished Name (DN), and a full, non-searchable definition of each LDAP entry. The LDAP entry table is used to return the requested attributes from an LDAP search once the search has been narrowed down to a specific entry or set of entries.

Tuning the DB2 buffer pools can have a dramatic effect on performance — in some cases as much as 10 times better. The performance improvement depends upon how much of the database can be cached. The more data that can be cached the greater the performance improvement.

Buffer pool tuning has less impact on directories containing tens of millions of users, since it is not possible or practical to cache a large enough percentage of the database to greatly improve performance. With directories up to millions of users, it is possible to cache a large enough percentage of the database to make an improvement to performance.

Determining the optimum size for the buffer pools can be difficult. If buffer pool sizes are set too high, system memory usage exceeds the available physical memory and the operating system spends much time paging memory.

One of the things that impacts memory usage is file system caching. The following paragraphs discuss when file system caching competes for memory and some options for turning file system caching off.

There are two types of database storage:

- System Managed Space (SMS)
- Database Managed Space (DMS)

In Tivoli Directory Server, versions 6.2 and later, DMS is the default table space type, and file system caching is disabled. In previous versions, the default table space type is SMS and file caching is enabled, which results in double caching. In double caching, one cache is the DB2 buffer pools; the other is the file system cache.

If DB2 buffer pools are the only cache, system performance improves. You can disable file system caching by using DMS on raw devices, or by setting options for file-based DMS or SMS containers. DMS can be more complex to manage than SMS, but it provides options to turn off file system caching. Not all versions of DB2 or all the supported operating systems support turning off file caching.

For example, you can turn off file system caching under the following conditions:

Example 1:

- The operating system is AIX 5.2 or later.
- The version of DB2 is 8.2 or later.
- The database is Enhanced Journaled File System (JFS2).

Example 2:

- The operating system is Windows (all versions).
- DB2 versions before 8.2.
- You enable the DB2NTNOCACHE environment variable.

The DB2 and Tivoli Directory Server documentation provides more information about what versions of DB2 and operating systems support turning off file system caching.

As mentioned, the DB2 buffer pools sizes must be low enough to prevent operating system paging, but high enough to provide maximum benefit. Another consideration is allocation of memory to the two buffer pools:

- IBMDEFAULTBP
- LDAPBP

The IBMDEFAULTBP is primarily an attribute cache and the LDAPBP is an LDAP entry cache.

The following results show the impact of buffer pool tuning for a directory containing 2 million Security Access Manager users in the minimal data format:

- 2M Security Access Manager users in the minimal data format
- AIX 5.2 Database on a JFS2 file system
- Database on a single disk
- DB2 version 8.2
- 2 GB of physical memory

This table shows a summary of the test parameters and results:

Table 2. Test parameters

Test #	**Total Buffer Pool Size**	**IBMDEFAULTBP** **Size**	**LDAPBP** **Size**	**File System Caching**	**Security Access Manager Authentication Throughput Rate**
1	259 MB	49800 pages of 4 KB 195 MB	2075 pages of 32 KB 65 MB	on	41 auths/sec

Test #	Total Buffer Pool Size	IBMDEFAULTBP Size	LDAPBP Size	File System Caching	Security Access Manager Authentication Throughput Rate
2	259 MB	49800 pages of 4 KB 195 MB	2075 pages of 32 KB 65 MB	off	22.3 auths/sec
3	1 GB	196608 pages of 4 KB 768 MB	8192 pages of 32 KB 256 MB	off	85.6 auths/sec
4	1 GB	131072 pages of 4 KB 512 MB	16384 pages of 32 KB 512 MB	off	127 auths/sec
5	1 GB	98304 pages of 4 KB 384 MB	20480 pages of 32 KB 640 MB	off	137 auths/sec
6	1 GB	65536 pages of 4 KB 256 MB	24576 pages of 32 KB 768 MB	off	74.3 auths/sec
7	259 MB	33200 pages of 4 KB 130 MB	4160 pages of 32 KB 130 MB	on	41.0 auths/sec

Notes:

- A Security Access Manager authentication consists of 3 LDAP operations.
- In all cases, the amount of file I/O during the test was high indicating that the entire database was not cached. Much higher performance is attained if the entire database is cached, which is only possible in this environment if more physical memory were available.
- The physical memory is close to fully used and paging rates are low, unless otherwise noted in the discussion here.

Test 1 shows an optimal buffer pool allocation with file system caching turned on.

Test 2 shows that performance degrades when file system caching is turned off and the buffer pool setting is left unchanged. For test 2 the physical memory usage is low, indicating that turning off the file system cache frees up a large amount of memory.

Test 3 shows a performance improvement from turning off file system caching when the buffer pool size is increased from 259 MB of memory to 1 GB of memory.

Tests 4, 5, and 6, shows the performance with various allocations of the 1 GB of memory divided between the IBMDEFAULTBP and the LDAPBP buffer pools. The optimum allocation in this case is 384 MB to the IBMDEFAULTBP and 640 MB to the LDAPBP.

Test 7 shows a different allocation of memory between the IBMDEFAULTBP and the LDAPBP buffer pools with file system caching turned on. No change to performance is seen with this different allocation as compared to test 1 which is also run with file system caching turned on.

In conclusion, a large performance gain is achieved by turning off file system caching. If file system caching cannot be turned off, the buffer pool is to be allocated as in test 1. If file system caching can be turned off, the results imply that for 2M Security Access Manager users and Security Access Manager authentication, performance is enhanced by allocating 384 MB to the IBMDEFAULTBP buffer pool (for caching attributes) and allocating the remainder 640 MB to the LDAPBP buffer pool (for caching LDAP entries).

A more general guideline with file system caching turned off is to allocate 384 MB / 2 million or about 200 bytes per Security Access Manager user to the IBMDEFAULTBP ,and the remaining available physical memory to the LDAPBP. Assume that 1 GB of physical memory is required for Tivoli Directory Server and DB2 process sizes.

Setting buffer pool parameters with the db2_tunings.sh script

The db2 tunings.sh script sets the buffer pool sizes to optimum values for a AIX systems with 2 GB of physical memory. Based upon the users response on a file system caching support question, the script sets the buffer pool sizes as follows:

Table 3. Buffer pool sizes

File System Caching Response	Total Buffer Pool Size	IBMDEFAULTBP Size	LDAPBP Size
supported	1 GB	98304 pages of 4 KB 384 MB	20480 pages of 32 KB 640 M
not supported	259 MB	49800 pages of 4 KB 195 MB	2075 pages of 32 KB 65 MB

If the user indicates that the file system caching option is supported by DB2, the script additionally issues the DB2 commands to turn off file system caching.

For other settings, edit the db2_tuning.sh script and search for the **ibmdefaultbp** and **ldapbp** to find example settings. Modify the script comments to enable or disable the appropriate lines to produce the wanted settings or add a new setting based on one of the examples.

Setting the buffer pool parameters using DB2 commands

About this task

To determine the current sizes of the DB2 buffer pools, execute these commands:

```
db2 connect to ldapdb2
db2 select varchar(bpname,20) as bpname,npages,pagesize from syscat.bufferpools
```

To determine the current file system caching option for each of the table spaces, execute these commands:

```
db2 get snapshot for tablespaces on ldapdb2 | egrep
'Tablespace name|File system caching'
```

To turn off file system caching with DB2 version 8.2 or later, and with operating systems and file system environments that support it, use this commands:

```
db2 connect to ldapdb2
db2 alter tablespace USERSPACE1 no file system caching
db2 alter tablespace LDAPSPACE no file system caching
db2 terminate
db2stop
db2start
```

To set the buffer pool sizes, use these commands:

```
db2 alter bufferpool ibmdefaultbp size <new size in 4096 byte pages>
db2 alter bufferpool ldapbp size <new size in 32768 byte pages>
db2 terminate
db2stop
db2start
```

For example, to set the buffer pool sizes to optimum values for a system with file system caching turned on and 2 GB of physical memory, use these commands:

```
db2 alter bufferpool ibmdefaultbp size 49800
db2 alter bufferpool ldapbp size 2075
db2 terminate
db2stop
db2start
```

These settings result in the following buffer pool memory allocations, which are optimal for an AIX system with 2 GB of memory and file system caching turned on.

Table 4. Buffer pool memory allocations

Total Buffer Pool	IBMDEFAULTBP Size	LDAPBP Size
259 MB	49800 4 KB pages or 195 MB	2075 32 KB pages or 65 MB

If these commands are executed while the directory server is running, the **db2stop** command fails with a message indicating there are still applications connected to the database. If this occurs, stop the directory server, then execute the following commands:

```
db2stop
db2start
```

Buffer pool analysis

About this task

The following procedure is used to perform buffer pool analysis. This procedure is also documented in the bufferpool_snapshot_analysis.README file from the tuning guide scripts.

Procedure

1. Turn on buffer pool monitoring.

   ```
   db2 update database manager configuration using DFT_MON_BUFPOOL ON
   db2stop
   db2start
   ```

2. Connect to the database.

   ```
   db2 connect to ldapdb2
   ```

3. Start the workload to be analyzed.
4. Reset the monitor data.

   ```
   db2 reset monitor all
   ```

5. Wait about a minute as statistics are gathered with the workload in progress.
6. Take a snapshot of the buffer pool statistics and process the output.

   The following command takes a snapshot of the current buffer pool statistics:

   ```
   db2 get snapshot for bufferpools on ldapdb2
   ```

   The following command takes a snapshot and reports the read times:

   ```
   db2 get snapshot for bufferpools on idsdb | awk '{
    if($1=="Bufferpool"&&$2=="name"){print   $0}
   if (index($0,"pool read time")){print "\t"$0}
   }'
   ```

   The following command takes a snapshot and reports the number of logical and physical reads:

   ```
   db2 get snapshot for bufferpools on idsdb | awk '{
   ```

```
 if($1=="Bufferpool"&&$2=="name"){print   $0}
if (index($0,"cal reads")){print "\t"$0}
}' | grep -v temp
```

The following command takes a snapshot and reports miss ratios:

```
db2 get snapshot for bufferpools on idsdb | grep -v temporary | awk '{
if($1=="Bufferpool"&&$2=="name"){print   $0}
if (index($0,"logical reads")){l=$NF;getline;p=$NF;
if (l==0){r=0}else{r=p/l};print "\tMiss ratio: "$3" "r}
}'
```

7. Tune the buffer pool sizes such that the IBMDEFAULTBP has a low read time, a low number of reads, and a low miss ratio, but do not exceed system physical memory size or reduce the LDAPBP size to less than 2075 pages of 32 KB.
   A higher miss ratio means there are a higher number of physical reads and a lower number of cache hits.
   Allocate the remaining physical memory to the LDAPBP as follows: With file system caching turned off:
   LDAPBP size = (<total physical memory> - 1GB (for IDS and DB2) - (IBMDEFAULTBP size) * 4096) / 32768
   With file system caching turned on:
   LDAPBP size = (<total physical memory> - 1.75GB (for IDS, DB2 and file system caching) - (IBMDEFAULTBP size) * 4096) / 32768

Warnings about buffer pool memory usage

If any of the buffer pool sizes are set too high, DB2 fails to start due to insufficient memory. If this occurs, there might be a core dump file, but typically there is no error message.

On AIX systems, the system error log might report a memory allocation failure. To view this log, enter:

```
errpt –a | more
```

Restoring a database that was backed up on a system, with buffer pool sizes that are too large for the target system, causes the restoration to fail.

See “Trouble shooting” for information about how to work around this problem.

If DB2 fails to start due to buffer pool sizes being too large, set the buffer pool sizes to lower values and restart DB2.

Tuning the DB2 transaction log size

DB2 transaction log space is defined by the following DB2 parameters:

- LOGFILSIZ
- LOGPRIMARY
- LOGSECOND
- NEWLOGPATH

The output from the following DB2 command includes the current setting of these parameters:

```
db2 get database configuration for ldapdb2 |
egrep 'LOGFILSIZ|LOGPRIMARY|LOGSECOND|NEWLOGPATH|Path to log files'
```

The transaction log parameters must be tuned to allow the transaction logs to grow to their maximum required size.The transaction log size is limited by the LOGFILSIZ, LOGPRIMARY, and LOGSECOND DB2 parameter values, and also by the available disk space in the directory specified by the NEWLOGPATH DB2 parameter.

The Tivoli Directory Server uses transaction log disk space for storing uncommitted DB2 transactions from directory update operations.

The transaction log size is limited both by DB2 parameter values and the available disk space. If the transaction logs exceeds the limit, due to the settings of the DB2 size parameter, the transaction is backed out using the information in the transaction logs, and the transaction fails. If the transaction logs exceed the limit, due to a lack of available disk space, the database becomes corrupted, and goes into an unusable state.

If the database becomes corrupted in this way, it is possible, but more complex, to issue DB2 commands to recover the database. Alternatively, the database can be restored from a backup or reloaded.

In general, regular backups help reduce the complexity of recovering from failures, but the best practice is to ensure there is enough disk space to allow the transaction logs to grow to the DB2 parameter limits.

Note: If the database becomes corrupted, often the recovery commands can be found in the sqllib/db2dump/db2diag.log file which is located in the DB2 instance owners home directory.

In normal run time use of the Tivoli Directory Server, the transaction log requirements are small. For the **bulkload** utility, the transaction log requirements are even smaller. Some run time directory operations increases the transaction log requirements for a short period.

Here are a few examples:

- The **ldapadd** or **ldapmodify** commands use an increasing amount of transaction log space as the number of multi-valued attributes added to a single LDAP entry in a single command grows, for example as in loading many members into a group.

  The transaction log requirements to load a 3M user group are around 300 MB.

  Note: The Tivoli Directory Server **bulkload** command with the (**-G**) option can be used as an alternative to using the **ldapadd** and **ldapmodify** commands to add many members to an existing group without using transaction log space.
- An ACL placed on a suffix object can result in that ACL propagating to every entry under that suffix.

  The Tivoli Directory Server performs ACL propagation as one single committed DB2 transaction.

  The transaction log requirements to propagate an ACL to a suffix with 3M Security Access Manager users in the standard (old) data format is around 1.2GB.

> The Security Access Manager fixacls.sh tuning guide script has the same transaction log requirement as ACL propagation, since it performs a similar function.

Using the 1.2GB requirement for the propagation of ACLs to 3M Security Access Manager users, the transaction log requirements are approximately:

1.2*1024MB / 3 million users = 410 bytes per user

By default the DB2 transaction log file size (LOGFILSIZ) is defined to be 2000 blocks of 4 KB in size or 8000 KB per log file. The number of primary logs files (LOGPRIMARY) is defined as 3 and the number of secondary log files (LOGSECOND) is 2.

The following formula shows the default transaction log space limit:

2000 * 4096 * (3 + 2) = 39.1MB

The default is sufficient for approximately 100 thousand users. In order to increase the DB2 transaction log limits to allow for millions of users, it is necessary to increase the size of the transaction logs (LOGFILSIZ), and increase the number of secondary files (LOGSECOND). It is better to increase the number of secondary files, rather than the number of primary files, because the secondary files periodically get deleted when not in use.

The db2_tuning.sh script can be used to adjust the DB2 transaction log parameters for the worst case of propagating ACLs. This script modifies the transaction log file size (LOGFILSIZE) to 10000 blocks of 4 KB in size or about 39 MB per log file. The script modifies the number of primary log files (LOGPRIMARY) to 2. The script prompts for the number of Security Access Manager users that are loaded into the directory server and sets the number of secondary logs (LOGSECOND) according to the following formula:

LOGSECOND = (<num AM users> * 410) / <39MB log file size> - 2 primary log file + 1

This transaction log setting can be performance manually using this DB2 commands:

```
db2 update database configuration for LOGFILSIZE using 10000
db2 update database configuration for LOGPRIMARY using 2
db2 terminate
db2 force applications all
db2 connect to ldapdb2
db2 get database configuration for ldapdb2 | \ egrep 'LOGFILSIZ|LOGPRIMARY|
LOGSECOND|NEWLOGPATH|Path to log files'
```

For Tivoli Directory Server, you can set the LOGFILSIZ and NEWLOGPATH parameters using the performance tuning tool **idsperftune**.

If the **db2 force applications all** command is issued while the Tivoli Directory Server is running, the server becomes partially functional. See the note at the beginning of this section for more details.

The db2_tuning.sh script verifies that the disk space in the directory specified in the NEWLOGPATH DB2 parameters is sufficiently large to allow the transaction logs to grow the limits defined by the DB2 parameters. Use the UNIX **df** command or check the properties of the folder on a Windows systems to do this manually.

Here is an example of the **df** command:

```
df -k /home/idsldap/idsldap/NODE0000/SQL00001/SQLOGDIR/
```

where /home/idsldap/idsldap/NODE0000/SQL00001/SQLOGDIR/ is the current setting of the NEWLOGPATH DB2 parameter.

The free disk space must be greater than or equal to the size defined by the following formula:
(LOGFILSIZ * 4096 * (LOGPRIMARY + LOGSECOND) / 1024)

This command and formula give the transaction log limits in kilobytes.

Improving performance with separate disks for transaction logs

Put the DB2 table spaces and transaction logs on different disks to improve performance. Put the containers for LDAPSPACE and USERSPACE1 table spaces on a separate physical disk from the DB2 transaction logs. Separating table spaces and logs requires three disks:

- Use two disks for the table spaces.
- Create containers for each table space on each disk.
- Use one disk for the transaction logs.

A DB2 backup and redirected restore are required to move the table spaces. To move the transaction log, use the DB2 NEWLOGPATH and LOGFILSIZ parameters. For details, see *IBM Tivoli Directory Server 6.2 Performance Tuning and Capacity Planning Guide*.

Tuning the number of database connections

The default number of LDAP to DB2 connections is 15. This number is sufficient for most environments. You might want to increase the number of back-end connections to 30, depending on the server load and the nature of concurrent independent connections.

About this task

To determine the right number of connections, look at the result of the monitor search.

Procedure

1. Run the following command:
   ```
   idsldapsearch -p 3389 -D cn=root -w root -s base -b cn=monitor objectclass=*
   ```
2. Look for available workers in the output. Database connections and worker threads tend to be synonymous.
3. To increase the worker threads and the number of possible connections, set the following parameter: *ibm-slapdDbConnections* under dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, cn=Configuration.

ISAM for Mobile/Federation External DB2 Tuning

About this task

This task describes how to tune DB2 server when used as an external run time High Volume Database (HVDB) for ISAM for Mobile/Federation.

1. Tuning I/O

We may hit the disk I/O bottleneck for a db2 server if we keep the DB2 logpath and the high volume transaction tables on the same disk, it is recommended to change transaction logpath to a separate disk.

By default the DB alias name is “HVDB”. In the following example we will use the default alias name for sample commands.

Command to check existing logfile path:

```
db2 get database configuration for HVDB
```

Snippet of sample output:

```
Default number of containers                          = 1
Default tablespace extentsize (pages)   (DFT_EXTENT_SZ) = 32

Max number of active applications           (MAXAPPLS) = AUTOMATIC(60)
Average number of active applications       (AVG_APPLS) = AUTOMATIC(1)
Max DB files open per application           (MAXFILOP) = 65535

Log file size (4KB)                        (LOGFILSIZ) = 1024
Number of primary log files               (LOGPRIMARY) = 13
Number of secondary log files              (LOGSECOND) = 12
Changed path to log files                 (NEWLOGPATH) =
Path to log files                                      = C:\DB2\NODE0000\SQL00001\LOGSTREAM0000\
Overflow log path                    (OVERFLOWLOGPATH) =
Mirror log path                        (MIRRORLOGPATH) =
First active log file                                  =
Block log on disk full               (BLK_LOG_DSK_FUL) = NO
Block non logged operations           (BLOCKNONLOGGED) = NO
Percent max primary log space by transaction  (MAX_LOG) = 0
Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0
```

Command to change logfile path:

```
db2 update db cfg for HVDB using NEWLOGPATH /newpath
```

where newpath is the path on the separate disk.

Restart the DB2 instance.

2. Increasing transaction logsize

The default setting for transaction logsize may not be sufficient when system is under heavy load. Change the default setting for transaction logsize:

1. Connect to the DB:

   ```
   db2 connect to HVDB
   ```

2. Get the correct log configuration (check if the current value and delayed value are equal)

   Windows:
   Type the command below and check for parameters logfilsiz, logprimary, logsecond:

   ```
   db2 get db cfg for HVDB show detail
   ```

   Unix:

   ```
   db2 get db cfg for HVDB show detail | grep -i LOG
   ```

3. Decide if the log needs to be increased.
   The parameter LOGFILSIZ shows the amount of 4k pages, so if is showing 1024 each log will be about 4MB. Multiply this by the number of primary and secondary logs, and you will have your maximum log size.

   Update your parameters
   You can choose between increasing one of the following parameters: logfilsiz, logprimary, logsecond

   LOGFILSIZ - Size of log files configuration parameter
   This parameter defines the size of each primary and secondary log file. The size of these log files limits the number of log records that can be written to them before they become full and a new log file is required. This size is measured in 4k pages.

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

   LOGSECOND - Number of secondary log files configuration parameter
   This parameter specifies the number of secondary log files that are created and used for recovery log files (only as needed).

   In the example below we are setting each log size to about 400MB, and 15 primary logs means that every time there will be at least 15 log files available to this database and a maximum of 45 log files available.

   ```
   db2 update db cfg for HVDB using logprimary 15

   db2 update db cfg for HVDB using logsecond 30

   db2 update db cfg for HVDB using logfilsiz 100000
   ```

4. Close your connection, connect again and confirm values of transaction logsize parameters:

   ```
   db2 terminate
   db2 force applications all
   db2 connect to DBNAME
   db2 get db cfg for DBNAME | egrep
    'LOGFILSIZ|LOGPRIMARY|LOGSECOND|NEWLOGPATH|Path to log files'
   ```

6. Creating the LDAP_DESC(AEID,DEID) index

Create the (AEID,DEID) index on the DB2 LDAP_DESC table. Creation of the LDAP_DESC(AEID,DEID) index primarily improves the performance of small subtree operations.

It is not required to stop the Tivoli Directory Server before you create the LDAP_DESC(AEID,DEID) index. Not stopping it might cause timeouts on the directory server. See "Stopping the Tivoli Directory Server" for details.

The following are some tuning examples:

- Performing a small subtree with an objectclass=* filter when 100 K or users exist in the directory server.
- Placing an ACL on a small subtree when millions of entries exist in the directory server, for example, when you use **ldapmodify** to place the ACL.
- Using the fixacls.sh tuning guide script.

  The performance benefit of the AEID,DEID index is at its greatest when the fixacls.sh script is run after a bulkload of a relatively small number of LDAP entries into a directory server with millions of LDAP entries.

  The fixacls.sh is needed only when:

  - The standard LDAP data format option is selected, and

  - 100 K or more users are loaded into the directory server by using **bulkload**

  The only data format model that is supported is the standard format. In Security Access Manager, the **amldap2v6** utility is provided to migrate an LDIF file dumped by using the **db2ldif** command to the new minimal data format. The minimal data format does not require the use of the fixacls.sh tuning script after a **bulkload**.

The last two cases are the result of propagating ACLs to the children of small subtree. Propagating ACLs to the children of a large subtree is slow regardless of the existence of this index.

Without the LDAP_DESC(AEID,DEID) index, all subtree searches by using an objectclass=* filter are slow on a large directory server, even if the subtree searched is small.

One thing to consider when you design an LDAP search is the order in which the subtree criterion is used in evaluating the response. The subtree criterion is used last. Because of this, subtree searches specifying a commonly used object class values on a small subtree of a large directory are slow even if the LDAP_DESC(AEID,DEID) exists. For best performance, avoid searches which specify a commonly used object class value.

If the directory server has many existing entries, the creation of this index takes a long time. One measure is 35 minutes on a directory server with 3M Security Access Manager users. To check for the existence of the LDAP_DESC(AEID,DEID) index use the following command:

```
db2 connect to ldapdb2
db2 describe indexes for table ldap_desc show detail
```

If the output does not include a line which contains +DEID+AEID, the index does not exist.

Table 5. DB2 indexes for table ldap_desc

Index schema	Index name	Unique rule	Number of columns	Column names
IDSLDAP	LDAP_DESC_AEID	D	I2	+DEID+AEID

Use the following command to create the LDAP_DESC(AEID,DEID) index:

```
db2 connect to ldapdb2
db2 "create index LDAP_DESC_DEID on LDAP_DESC(AEID,DEID)"
```

Use the following command to drop the LDAP_DESC(AEID,DEID) index:

```
db2 connect to ldapdb2
db2 drop index IDSLDAP.LDAP_DESC_DEID
```

7. Tuning the Tivoli Directory Server change log

The change log is used to track updates. Security Access Manager does not use the change log functionality.

Before performing this task, make sure the Tivoli Directory Server is started.

The Tivoli Directory Server server change log significantly slows down update performance. The change log causes all directory updates to be recorded in a separate change log DB2 database, separate from the Tivoli Directory Server database.

The change log is used to track updates. IBM Security Access Manager does not use the change log functionality.

To determine the change log configuration status, search for the CN=CHANGELOG pseudo suffix as follows:

```
ldapsearch -h ldap_host  -D cn=root -w ldap_password  -s base -b “”
"objectclass=*" | grep "CN=CHANGELOG"
```

where *ldap_password* is the password for the directory administrator.

If there are no results presented, the Change Log option for Tivoli Directory Server is not configured.

This is the default setting for Tivoli Directory Server, Version 6.3.

To manage change log, start idsxcfg and **Manage changelog**.

8. Tuning the Tivoli Directory Server audit log

Before performing this task, make sure that the Tivoli Directory Server is started.

The Tivoli Directory Server audit logging feature significantly slows down many aspects of the Tivoli Directory Server server performance, depending upon which audit logging features are turned on.

Turn off all audit logging features.

To check the status of the audit logging feature, run the following **ldapsearch** command:

```
ldapsearch –D cn=root –w ldap_passwd  –s base –b
cn=audit,cn=localhost "objectclass=*"  ibm-audit
```

where cn=root is the Tivoli Directory Server root administrator user, and *ldap_passwd* is the password for this administrator.

An example output from this command is, as follows:

```
cn=audit,cn=localhost
ibm-audit=false
```

where ibm-audit=false indicates that audit logging is turned off. If this value is true, issue the following command to turn it off:

```
cat <<EOF | ldapadd -D cn=root -w ldap_passwd
dn: cn=audit,cn=localhost
changetype: modify
replace: ibm-audit
ibm-audit: false
EOF
```

For Tivoli Directory Server, Version 6.3, use the following command to check the audit logging status:

```
ldapsearch –D cn=root –w ldap_passwd  –s base –b cn=audit,
cn=Log Management,cn=configuration"
"objectclass=*" ibm-audit
```

where cn=root is the Tivoli Directory Server root administrator user, and *ldap_passwd* is the password of this administrator.

An example output from this command is, as follows:

```
cn=Audit,cn=Log Management, cn=Configuration
ibm-audit=false
```

If this value is true, issue the following command to turn it off:

```
idsldapmodify -p <port> -D <adminDN> -w <adminPwd>
```

```
dn: cn=Audit, cn=Log Management, cn=Configuration
changetype: modify
replace: ibm-audit
ibm-audit: false
```

9. Configuring the Tivoli Directory Server suffixes

Before you can add any LDAP entry to Tivoli Directory Server, you must configure at least one suffix. In a Security Access Manager environment, the secauthority=default suffix is typically configured.

About this task

For optimum Security Access Manager authentication performance, define one other suffix to hold all Security Access Manager user LDAP entries.

Security Access Manager authentication performs more searches to determine under which suffix a user LDAP entry exists when:

- More than one suffix is defined.
- Security Access Manager users are under more than one suffix.

Security Access Manager uses the most frequently used suffix algorithm to determine which suffix to search first. More searches are required for the authentication of a user with an infrequently used suffix. Instead of placing users in multiple suffixes, place users in multiple LDAP container entries under the same suffix for better performance.

To determine what suffixes are currently defined, run a command similar to the following one:

```
ldapsearch -L -D cn=root -w <password> -s base \
-b "cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, \
cn=Configuration" "objectclass=*" ibm-slapdSuffix
```

In addition to any user created suffixes, the output from the command, which is shown here, always includes the following suffixes:

```
ibm-slapdSuffix=cn=localhost
ibm-slapdSuffix=cn=ibmpolicies
ibm-slapdSuffix=cn=Deleted Objects
```

Procedure

1. Make sure that the Tivoli Directory Server is running.
2. Repeat the following command for each suffix that you want to configure:
   ```
   cat << EOF | ldapmodify ldap_credential
   dn: cn=Directory, cn=RDBM Backends, cn=IBM Directory, cn=Schemas, \
   cn=Configuration
   changetype: modify
   ```

```
add: ibm-slapdSuffix
ibm-slapdSuffix: suffix_name
EOF
```

3. Take one of the following actions:
   - Restart the Tivoli Directory Server.
   - Run the following command to make the newly configured suffixes known to the Tivoli Directory Server:

     ```
     ldapexop -D ldap_credential  -op readconfig -scope single \
     "cn=Directory,cn=RDBM Backends,cn=IBM Directory, \
     cn=Schemas,cn=Configuration" ibm-slapdSuffix
     ```

     Where *ldap_credential* are the **ldapsearch** parameters to identify the LDAP server and LDAP user to do the update. *suffix_name* is the LDAP suffix to configure.

Results

For Security Access Manager, use the command which is shown to configure the secauthority=default and one or more user suffixes. For example, c=us or o=mycompany.com.

10. Creating LDAP entries for configured suffixes

After configuring the Tivoli Directory Server suffixes, the LDAP entries that define each suffix must be created.

The following are some examples of how the LDAP entries on suffixes can be created:

- Use the **ldapadd** command to create the LDAP entries for suffixes.
- Restore a directory database containing the LDAP entries for suffixes.
- Load the directory server with an LDIF file containing the LDAP entries for suffixes.

It is not necessary to create the secauthority=default LDAP entry. The secauthority=default suffix is created by Security Access Manager configuration.

Using the **ldapadd** command, the following example creates a suffix object named o=ibm.com:

```
cat <<EOF | ldapadd -h ldap_host -D cn=root -w ldap_passwd
dn: o=ibm.com
objectClass: organization
objectclass: top
o: ibm.com EOF
```

where cn=root is the Tivoli Directory Server server root administrator user, *ldap_host* is the host name of the Tivoli Directory Server server, and *ldap_passwd* is the Tivoli Directory Server root administrators password.

This example does not assign any access control lists (ACLs) to the suffix object. You can assign any ACL. However, keep in mind that Security Access Manager adds its own ACLs to these objects and enables ACL propagation. Security Access Manager makes these ACL changes to allow its servers to access all objects in the directory tree.

If an ACL design is chosen that conflicts with Security Access Manager's ACL model, Security Access Manager cannot operate on the affected user objects.

11. Loading Security Access Manager users

Prepare Tivoli Directory Server for loading Security Access Manager users. Before performing this task, make sure that the Tivoli Directory Server is started.

Before loading Security Access Manager users into the Tivoli Directory Server, the Security Access Manager directory tree must be created on the directory server.

The Security Access Manager directory tree includes LDAP container entries for holding Security Access Manager user and group information.

The easiest way to create the Security Access Manager directory tree is to configure Security Access Manager into the Tivoli Directory Server. If this is done, it is also not necessary to apply the Security Access Manager schema to the directory server.

It is sometimes convenient to load Security Access Manager users into the Tivoli Directory Server before Security Access Manager has been configured. In this case, use the Security Access Manager tuning scripts as described in the following sections to create an unconfigured Security Access Manager directory tree.

Applying the Security Access Manager schema to the Tivoli Directory Server

To add Security Access Manager objects to the Tivoli Directory Server server, the Security Access Manager schema must be applied to the Tivoli Directory Server server.

Enter the following **ldapadd** command to accomplish this task:

```
ldapadd -h ldap_host  -D cn=root -w ldap_passwd  -f secschema.def
```

where cn=root is the Tivoli Directory Server server root administrator user, *ldap_host* is the host name of the Tivoli Directory Server server, and *ldap_passwd* is the Tivoli Directory Server server root administrators password.

The secschema.def file can be found either in the /opt/PolicyDirector/etc directory on a system with the Security Access Manager installed or with the scripts described.

If a previous version of the Security Access Manager schema has already been added to the Tivoli Directory Server server, a different schema definition file must be used to upgrade the schema, instead of adding it.

For more information, see the *IBM Security Access Manager: Installation Guide*.

Creating an unconfigured Security Access Manager directory tree

To create an unconfigured Security Access Manager directory tree, enter the following command:

ldapadd -h *ldap_host* -D cn=root -w *ldap_passwd* -f am_min_unconfig.ldif

where

- cn=root is the Tivoli Directory Server server root administrator user.
- *ldap_host* is the host name of the Tivoli Directory Server server.
- *ldap_passwd* is the Tivoli Directory Server server root administrators password.
- am_min_unconfig.ldif is the name of the file containing the LDAP Information File (LDIF) input for defining the Security Access Manager objects.

Create an unconfigured Security Access Manager subdomain tree

Create an unconfigured Security Access Manager subdomain if users are to be loaded into a subdomain before Access Manager is configured into the Directory server.

Refer to the *IBM Security Access Manager: Administration Guide* for more information about Security Access Manager subdomains.

To create the subdomain object space, run the following tuning script piped to the **ldapadd** command:

mk_am_min_unconfig_dom_ldif.sh*dom* | ldapadd *ldap_cred*

where *dom* is name of the subdomain and *ldap_cred* is the LDAP credential. For example:

```
-D cn=root -w myldappaswd
```

To configure the domain after it has been created in this way, use the following command:

```
pdadmin domain create
```

12. Tuning the Tivoli Directory Server ACLs

To prevent performance delays when placing ACLs on Tivoli Directory Server suffixes, place ACLs on the suffixes when there are few or no users or LDAP entries under the suffix.

Before performing this task, make sure that the Tivoli Directory Server is started.

Performance delays and large transaction log space requirements occur when LDAP Access Control Lists (ACLs) are placed on Tivoli Directory Server suffixes and the following conditions are true:

- Millions of LDAP entries exist under the suffix.
- There is no previous ACL on the suffix.

An ACL can be placed on a Tivoli Directory Server using one of the following procedures:

- Configure Security Access Manager into the Tivoli Directory Server.
- Follow the steps in the *IBM Security Access Manager: Administration Guide* for manually creating the Tivoli Directory Server ACLs.
- Use the Security Access Manager check_ldap_acls.sh tuning script to automate the manual steps from the *IBM Security Access Manager: Administration Guide*.
- Refer to the Tivoli Directory Server to place an arbitrary ACL on all suffixes.

The last procedure prevents performance delays, but might apply ACLs that do not sufficiently protect the directory server or that do not give Security Access Manager enough permission to perform all authentication and administration functions. Use the last procedure if one of the other procedures is performed later, for example if Security Access Manager is later configured into the Tivoli Directory Server.

If one of these procedures is performed when millions of users exist under the suffix and there is no previous ACL on the suffix, performance delays occur.The following information describes the use of the check_ldap_acls.sh tuning script.

Run this script under the context of any user. The file system owner and the group must enable the user permission to run the file. Use this script to check ACLs by running it as follows:

```
check_ldap_acls.sh ldap_host ldap_passwd
```

where *ldap_host* is the host name of the Tivoli Directory Server server, and

ldap_passwd is the Tivoli Directory Server server root administrators password.

To use this script to add the missing ACLs reported on the check, enter on one line:

```
check_ldap_acls.sh ldap_host  ldap_passwd  | ldapadd -h ldap_host
-D cn=root -w ldap_passwd
```

where cn=root is typically the Tivoli Directory Server server root administrator user.

13. Tuning the DB2 indexes

DB2 indexes primarily improve the performance of Tivoli Directory Server search operations on the attributes that are indexed. The Tivoli Directory Server comes with a set of attributes that are indexed by default.

The definition of which attributes are indexed by default is found in the directory schema files. An attribute with the keyword EQUALITY in the schema files indicates that the attribute is indexed. Attributes can also be indexed issuing DB2 commands on the DB2 table that implement the attribute.

The following DB2 commands describe the indexes on the **postaladdress** attribute table:

```
db2 connect to ldapdb2
db2 describe indexes for table postaladdress show detail
```

The following DB2 commands create an index on the **postaladdress** attribute:

```
db2 connect to ldapdb2
db2 create index on postaladdress
db2 "create index postaladdress on postaladdress(postaladdress_t,eid)"
db2 runstats on table postaladdress with distribution and detailed indexes all
```

It is important to issue the **runstats** command on a table after an index is created. The **runstats** command provides statistical information to the DB2 optimizer that aids the optimizer to decide about optimizing searches on that table.

DB2 indexes also improve the performance of directory operations such as:

- Start time
- Finding the subtree of an LDAP entry
- Finding all children of an LDAP entry

In some versions of Tivoli Directory Server, the DB2 indexes definitions are not at their optimal settings.

Sometimes a database administrator might drop or redefine an index to be suboptimal. The Security Access Manager check_db2_indexes.sh tuning script can be used to check the DB2 index definitions and automatically correct them if wanted. For each index that is not optimal, the script reports the following and prompts for permission before any updates:

- Current setting
- Optimum setting

To check the DB2 indexes, run the check_db2_indexes.sh script without any parameters.

The **db2look** command is useful for reporting of all the tables and indexes in database. Use the Extract DDL option (**-e**) to produce a report that contains the DB2 commands that reproduce the current table and index definition.

For example:

```
db2look -e -d ldapdb2 -u ldapdb2 -o output_file
```

Where *output_file* is the file location for storing the results.

Before you run the **db2look** command, switch the user context to the database instance owners account.

You can reorganize DB2 indexes with the **idsdbmaint** tool. Use the **–i** argument. See the *Tivoli Directory Server Performance Tuning Guide* for more details.

14. Updating the DB2 system statistics

Although it is not required to stop the Tivoli Directory Server before performing this task, not doing so could result in timeouts when using the directory server. See "Stopping the Tivoli Directory Server"

The DB2 optimizer uses DB2 system statistics and the volatile table definitions to optimize DB2 queries. When LDAP entries are added to the directory server, for example when thousands of users are loaded into the directory, the DB2 statistics on the affected tables become out of date. The out of date system statistics can result in the DB2 optimizer making wrong choices that result in poor performance.

Forgetting to update system statistics is one of the most common performance tuning mistakes made with DB2 databases.

The first step in updating the DB2 system statistics is to either run the DB2 **runstats** command on every table or run the DB2 **reorgchk** command. Newer versions of DB2 offer a way to automatically check for out of date system statistics and automatically update the out of date system statistic using the **runstats** command.

Do not use this automatic **runstats** option with the Tivoli Directory Server and Security Access Manager since updating the DB2 system statistics for the Tivoli Directory Server and Security Access Manager also includes overriding the default DB2 system statistics from the **runstats** command.

The final step of updating the DB2 system statistics is defining all tables as volatile. The following sections provide more information about these steps.

The Security Access Manager **tam_runstats.sh** tuning script provides an automatic way to update the DB2 system statistics. Execute the **tam_runstats.sh** script without arguments and prompt for inputs as necessary.

The following sections provide information about how to manually perform the updates performed by the **tam_runstats.sh** script.

Updating system statistics with DB2 runstats or reorgchk commands

Update the DB2 system statistics by running either **runstats** on all the tables in the database or **reorgchk**.

Run the following commands to view the list of tables in the database:

- **db2 connect to ldapdb2**
- **db2 list tables for all**

Run the DB2 **runstats** command only on the tables that are listed with a DB2 instance schema. In this document, the DB2 instance schema is **ldapdb2**. Run the DB2 **runstats** command as follows:

For DB2, version 8 and later:

```
db2 runstats on table tablename  with distribution and detailed indexes all
allow write access
db2 runstats on table tablename with distribution and detailed indexes all
shrlevel change
```

Where *tablename* is the name of the table on which to run the **runstats** command.

The shrlevel change and allow write access options make the database accessible to the Tivoli Directory Server while **runstats** is in progress. The options can cause poor performance and timeouts in the directory server operations.

Running the **runstats** command on all tables can be tedious if you do not automate it with a script. The alternative is the DB2 **reorgchk** command. Run the DB2 commands as follows:

- **db2 connect to ldapdb2**
- **db2 reorgchk update statistics on table all**

The **reogchk** command does more checking and reporting on the organization of the database. This information is only useful if the database is read sequentially. Typically, the Tivoli Directory Server database is accessed randomly.

The main advantage of the DB2 **runstats** command is the ability to select which tables to tune.

The Security Access Manager tam_runstats.sh script limits **runstats** to tables that are less than 100,000 rows. In most cases, tables with 100,000 rows do not require tuning.

Limit the **runstats** command into small tables to decrease the **runstats** time.

For example, if you add:

- A million users to a Tivoli Directory Server that already has a million users, the affected tables might have a million rows. Running the **runstats** command on any table might be unnecessary.
- A second million users and introduce a new LDAP attribute that the first million did not have, run the runstats command on the tables for the new attributes. Running the **runstats** command on any table might be unnecessary.

In both cases, the time to update the system statistics by using the **runstats** command is less than the time it takes to run the runstats command on the tables in the directory database.

Example of the **runstats** performance:

It takes 20 minutes to run the **runstats** command on all tables of a 3 million Security Access Manager-user Tivoli Directory Server on a Sun Solaris 400 MHz system.

Updating system statistics with idsrunstats

You can run the **idrunstats** tool in Tivoli Directory Server 6.2 release while the directory server is running. See the *Tivoli Directory Server 6.2 Performance Tuning Guide* for details.

Improving disk utilization with DB2 row compression

If you have purchased a fully licensed version of DB2 Enterprise Server Edition, in addition to a DB2 Storage Optimization Feature licence, you will be able to use row-level compression.

You can improve the disk utilization and, in some cases, overall performance with the DB2 row compression capability. You can select the rows you want to compress with the **idsdbmaint** tool in the Tivoli Directory Server 6.2. See the *Tivoli Directory Server 6.2 Performance Tuning Guide* for details.

Defining DB2 tables as volatile

DB2 versions 8 and above support the option to define tables as volatile. By doing so, the DB2 optimizer always uses the indexes defined on the table, even if the system statistics indicate the table is small. The advantage to tuning the DB2 optimizer in this way is the potential that the DB2 optimizer makes the correct optimization choice when the statistics are out of date, for example when a table suddenly grows in size before the system statistics can be updated.

Run the following command to define a table as volatile:

```
db2 connect to ldapdb2 db2 alter table tablename volatile
```

Where *tablename* is the name of the table to define as volatile.

It can be tedious to perform this step manually. To automate the task use the Security Access Manager tam_runstats.sh script or the db2_volatile_tune.sh script. The tam_runstats.sh script performs all three steps for updating the DB2 system statistics.

Overriding system statistics

In some cases, the DB2 optimizer makes a wrong choice in optimizing a query, even if the system statistics are up to date. For those cases, it is necessary to override the DB2 system statistics to influence the DB2 optimizer to make the correct choice. Some of the system statistic overrides are performed by the Tivoli Directory Server, for example when the directory server is started and as part of the **idsrunstats** command.

Some of the system statistic overrides are unique to Security Access Manager. The system statistic overrides for both the Tivoli Directory Server and Security Access Manager are performed by the Security Access Manager sysstat_tune.sh script. The tam_runstats.sh script performs all three steps for updating the DB2 system statistics.

Run the sysstat_tune.sh script without parameters to override the DB2 default system statistics for Tivoli Directory Server and Security Access Manager. The sysstat_tune.sh script can be run with a disable parameter to return the system statistics to their default settings.

To control when the Tivoli Directory Server performs system statistic overrides, the LDAP_MAXCARD environment variable can be used.

For more information about setting LDAP_MAXCARD environment variable and its behavior, search the IBM support Web page for "LDAP_MAXCARD:

http://www.ibm.com/support

LDAP_DESC

This override sets the cardinality column of the LDAP_DESC table in the system statistic table to the value of 9E18, which is scientific notation for a large number. This override is also performed by the Tivoli Directory Server.

To perform this tune manually, run the following command:

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

This override causes the LDAP_DESC table to be chosen last when evaluating an LDAP search. The LDAP_DESC table is used on subtree searches when the LDAP_DESC(AEID,DEID) index is defined.

If the LDAP_DESC(AEID,DEID) index is not defined, this override has no effect. This override only allows small subtree searches to be fast if they are specified with an objectclass=* filter.

The main purpose of this override is to prevent the LDAP_DESC(AEID,DEID) index from being used on large subtrees by making the attributes specified in the search filter chosen first when evaluating an LDAP search.

LDAP_ENTRY

This override sets the cardinality column of the LDAP_ENTRY table in the system statistic tables table to the value of 9E18, which is scientific notation for a large number. This override is not performed by the Tivoli Directory Server.

To perform this tune manually, run the following command:

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

This override causes the LDAP_ENTRY table PEID index to be chosen last when evaluating an LDAP search. The PEID index is used to perform one level search. This is like the LDAP_DESC table override. It makes the attributes on the filter be used first when evaluating an LDAP search.

SECAUTHORITY

This override sets the cardinality column of the SECAUTHORITY table in the system statistic tables table to the value of 9E10, which is scientific notation for a large number. This override is not performed by the Tivoli Directory Server.

To perform this tune manually, run the following command:

```
db2 "update sysstat.tables set card = 9E10 where tabname = ’SECAUTHORITY’"
```

This override causes the SECAUTHORITY table to be chosen before the subtree criteria (LDAP_DESC table) and the one level criteria (LDAP_ENTRY table), but after all other attributes specified in an LDAP search filter. This improves the performance of Security Access Manager searches that specify the SECAUTHORITY attribute.

Specifically, this override improves the start time of the Security Access Manager servers when there are millions of users in the directory server. It causes the objectClass attribute to be used before the secAuthority attribute in narrowing down the entries that satisfy one of the searches that occur on start-up.

The SECAUTHORITY attribute is associated with Security Access Manager users and indicates which Security Access Manager domain the user is a member. If Security Access Manager subdomains are not used, the SECAUTHORITY attribute takes on the single value of DEFAULT.

CN

This override sets the cardinality column of the CN table in the system statistic tables table to the value of 9E10, which is scientific notation for a large number. This override is not performed by the Tivoli Directory Server.

To perform this tune manually, run the following command:

```
db2 "update sysstat.tables set card = 9E10 where tabname = ’CN’"
```

This override causes the CN table to be chosen before the subtree criteria (LDAP_DESC table) and the one level criteria (LDAP_ENTRY table), but after all other attributes specified in an LDAP search filter.

This override also allows the OBJECTCLASS attribute to be used before the CN attribute when evaluating an LDAP search. This improves the performance of some group searches performed by Security Access Manager.

REPLCHANGE

This override sets the cardinality column of the REPLCHANGE table in the system statistic tables table is to the value of 9E18, which is scientific notation for a large number. It also sets the colcard and high2key columns of the REPLCHANGE table with a colname of ID in the system statistic columns table to 9E18 and 2147483646.This override is also performed by the Tivoli Directory Server.

To perform this tune manually, run the following commands:

```
db2 "update sysstat.tables set card = 9E18 where tabname = ’REPLCHANGE’"

db2 "update sysstat.columns set colcard=9E18, high2key=’2147483646’ where

colname = ’ID’ and tabname = ’REPLCHANGE’"
```

This override allows DB2 to use the index defined on the REPLCHANGE table. Since the REPLCHANGE table varies greatly in size and is often empty, the system statistics when **runstats** typically catch the table when it is empty. The DB2 optimizer does not use indexes on what it thinks is an empty table.

For DB2 version 8 and later, this override is redundant with defining the table as volatile.

db2look command

The **db2look** command is useful for reporting of all the system statistic settings in the database. Use the mimic option (**-m**) to produce a report that contains the DB2 commands that reproduce the current system statistic settings.

For example:

```
db2look -m -d ldapdb2 -u ldapdb2 -o output_file
```

where *output_file* is the file location for storing the results.

Before running the ***db2look*** command, switch the user context to the database instance owners account.

15. Backing up the Tivoli Directory Server

In order to preserve the tuned Tivoli Directory Server, back it up.

Refer to "1. Backing up the Tivoli Directory Server" for more information about backing up the directory server.

16. Starting Tivoli Directory Server

Before the Tivoli Directory Server can be used, it must be started.

Refer to the section on "3. Starting the Tivoli Directory Server" for information about how to determine the current state of the Tivoli Directory Server and how to start it.

17. Testing the Tivoli Directory Server performance

After tuning the Tivoli Directory Server, test its performance using the Security Access Manager test_registry_perf.sh tuning script. Execute the script without any parameters to display the following usage information:

Usage: test_registry_perf.sh *user_principalname* *user_password* [*domain*]

where *user_principalname* is the name of a Security Access Manager user,

user_password is the user password, and *domain* is an optional parameter identifying the Security Access Manager domain to search. If the *domain* argument is not specified, the Security Access Manager default domain is searched.

The script prompts for the directory server instance name to use if more than one exists and it prompts for the root (cn=root) directory server administrator password.

The script performs **ldapsearch** commands like those performed by Security Access Manager during user authentication.Due to the operating activity of the **ldapsearch** command, the script can take up to 2 seconds to complete when the Tivoli Directory Server is fully tuned.

If the script takes longer than 2 seconds, it could be due to a cold DB2 buffer pool cache (not warmed up).Run the script a second or third time.

If the script continues to report greater than 2 second response times, investigate the problem by measuring individual commands reported on the script output and perform DB2 statement monitoring on commands that are slow.

Refer to "DB2 statement monitoring" for more information.

18. Tools

LDAP and DB2 analysis

Two developerWorks articles provide assistance with analyzing DB2 and LDAP performance. You can find these at the following URLs:

- http://www.ibm.com/developerworks/tivoli/library/t-tds-perf/
- http://www.ibm.com/developerworks/tivoli/library/t-tds-perf2/

Tivoli Directory Server Tools

Tivoli Directory Server has a number of useful tools. These are documented in the *Tivoli Directory Performance Tuning Guide*.

Following is a list of the tools that have been updated:

- **idsrunstats**
  This tool enables collection of DB2 system statistics. This is helpful if you want to tune Security Access Manager 7.0 and used tam_runstats.sh script in the past.
- **Idsperftune**
  You can use this tool to perform a number of tuning activities, such as providing suggestions, and updating parameters, based on user provided settings. It can also be used to perform more advanced tuning of DB2 parameters.
- **Idsdbmaint**
  You can use this tool to perform the following:
  - index reorganization.
  - row compression (see “Improving disk utilization with DB2 row compression”).
  - table space conversion (from SMS table space to DMS table space and DMS table space to SMS table space).

Chapter 3. Additional Tivoli Directory Server tuning

This chapter describes additional tuning procedures for Tivoli Directory Server.

Security Access Manager user data format

IBM Security Access Manager, version 9.0, provides two data formats for representing and storing Security Access Manager users in the Tivoli Directory Server: standard and minimal.

The two formats are mutually exclusive. All users must be defined in either one format or the other. If there is a mix of users in each of the format, Security Access Manager recognizes them in only one of the two formats. It depends on the version attribute in the secauthority=default LDAP entry. If the version is 6.0, Security Access Manager uses the new minimal data format. Otherwise, it uses the standard format.

Standard user data format

With the standard format, each Security Access Manager user is defined in a minimum of four LDAP entries. One entry is for the LDAP definition of the user and includes the user password. The remaining three LDAP entries contain Security Access Manager information for the user.

Two of the three Security Access Manager specific user LDAP entries are located as child entries under the LDAP user entry. To protect these two entries from unauthorized access, LDAP ACLs are on the Security Access Manager LDAP entry that is a direct descendant of the LDAP user entry.

The standard format requires one LDAP ACL per Security Access Manager user.

Tivoli Directory Server, version 6.2, and **bulkload** improve ACL performance. However, the extra time to load ACLs makes turning off **bulkload** processing an option to consider. The LDIF definition of the LDAP entries that are required to represent a user is one way to illustrate this format.

Use the Security Access Manager mk_test_users_ldif.sh and add_am_to_testusers_ldif.sh tuning scripts to generate the LDIF definition:

```
mk_test_users_ldif.sh 1 1 cn=testuser,o=ibm,c=us testpassword |
add_am_to_testusers_ldif.sh Default old acls
```

The following output is the result:

```
dn: cn=testuser1,o=ibm,c=us
objectclass: inetOrgPerson
objectclass: ePerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: testuser1
sn: test
userpassword: testpassword
uid: testuser1
```

```
dn: secAuthority=Default,cn=testuser1,o=ibm,c=us
objectclass: secUser
objectclass: eUser
objectclass: cimManagedElement
objectclass: top
secAuthority: Default
secLoginType: Default:LDAP
secPwdValid: true
principalName: testuser1
secUUID: 9077caee-9da1-11da-8be4-0006296cdc99
secHasPolicy: false
secPwdLastChanged: 20151208214816.0Z
secAcctValid: true
secDomainId: Default%testuser1
aclpropagate: TRUE
aclentry:group:CN=REMOTE-ACL-
USERS,CN=SECURITYGROUPS,SECAUTHORITY=DEFAULT:nor
mal:rsc:at.secAcctValid:rwsc:at.secPwdFailCountTime:rwsc:at.secPwdFailures:r
wsc:at.secPwdLastChanged:rwsc:at.secPwdLastFailed:rwsc:at.secPwdLastUsed:rws
c:at.secPwdUnlockTime:rwsc:at.secPwdValid:rwsc
aclentry: group:CN=IVACLD-
SERVERS,CN=SECURITYGROUPS,SECAUTHORITY=DEFAULT:norma
l:rsc:at.secAcctValid:rwsc:at.secPwdFailCountTime:rwsc:at.secPwdFailures:rws
c:at.secPwdLastChanged:rwsc:at.secPwdLastFailed:rwsc:at.secPwdLastUsed:rwsc:
at.secPwdUnlockTime:rwsc:at.secPwdValid:rwsc
aclentry: group:CN=SECURITYGROUP,SECAUTHORITY=DEFAULT:object:ad:normal:rwsc:se
nsitive:rwsc:critical:rwsc

dn: secUUID=9077caee-9da1-11da-8be4-0006296cdc99,cn=Users,secAuthority=Default
objectclass: secMap
objectclass: top
secDN: cn=testuser1,o=ibm,c=us
secUUID: 9077caee-9da1-11da-8be4-0006296cdc99

dn: cn=PolicyData,secAuthority=Default,cn=testuser1,o=ibm,c=us objectclass:
secPolicyData
objectclass: top cn: PolicyData
secPwdLastChanged: 20151208214816.0Z
```

The following summarized, hierarchical view of the output, shows some key object class definitions, attributes, and ACL placement:

```
dn: c=us,o=ibm
aclentry: ...ISAM ACLs...
 dn: cn=testuser1
 objectclass: inetOrgPerson
 userpassword: testpassword
  dn: secAuthority=Default
  objectclass: secUser
  secDomainId: Default%testuser1
  aclentry: ...ISAM ACLs...
  aclpropagate: TRUE
    dn: cn=PolicyData
    objectclass: secPolicyData
    secPwdLastChanged: 20151208214816.0Z

dn:secAuthority=Deault
```

```
objectclass: secAuthInfo
aclentry:...ISAM ACLs...
 dn: cn=Users
dn: secUUID=9077caee-9da1-11da-8be4-0006296cdc99
objectclass: secMap
```

Minimal user data format

The minimal user data format entries that are required reduces the number of LDAP entries needed to represent a Security Access Manager user from four to two.

As with the standard format, one entry is for the LDAP definition of the user and includes the user password. The remaining LDAP entry contains the Security Access Manager information for the user.

Another minimal data format enhancement moves the Security Access Manager-specific user LDAP entries from child entries to entries under the secauthority=default suffix. This move makes an LDAP ACL per Security Access Manager user unnecessary and eliminates the need for the fixacls.sh script. In the minimal format, Security Access Manager user LDAP entries are protected from unauthorized access by ACLs that are inherited from secAuthInfo LDAP entry (for example, secauthority=default) under which they are children.

The minimal format has the following performance benefits:

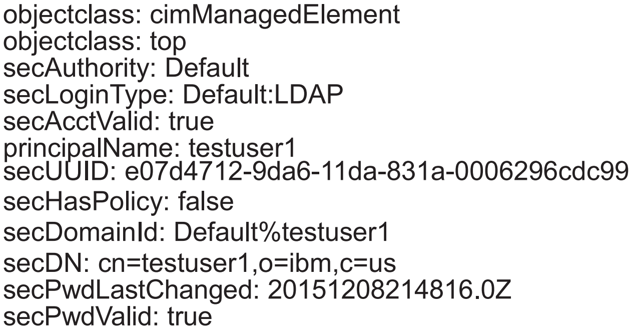
- Faster time to bulkload users because of smaller number of entries and the elimination of the fixacls.sh script.
- Faster time to administer users (create, change, delete) by using the **pdadmin** command and Java admin APIs.
- Reduced buffer pool foot print requires a smaller per user buffer pool cache. If the reduced footprint caches a large enough percentage of the users, it results in authentication performance gains.

The LDIF definition of the LDAP entries that are required to represent a user illustrates this format. Use the Security Access Manager mk_test_users_ldif.sh and add_am_to_testusers_ldif.sh tuning guide scripts to generate the LDIF definition as follows:

```
mk_test_users_ldif.sh 1 1 cn=testuser,o=ibm,c=us testpassword |
add_am_to_testusers_ldif.sh Default new
```

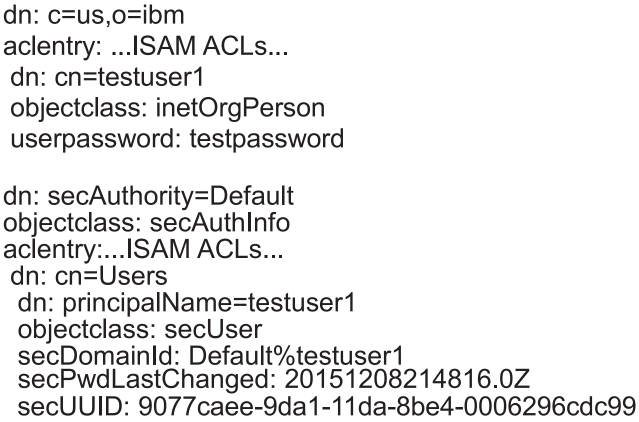
The following output is the result:

```
dn: cn=testuser1,o=ibm,c=us
objectclass: inetOrgPerson
objectclass: ePerson
objectclass: organizationalPerson
objectclass: person
objectclass: top
cn: testuser1
sn: test
userpassword: testpassword
uid: testuser1
dn: principalName=testuser1,cn=Users,secAuthority=Default
objectclass: secUser
objectclass: eUser
```



```
objectclass: cimManagedElement
objectclass: top
secAuthority: Default
secLoginType: Default:LDAP
secAcctValid: true
principalName: testuser1
secUUID: e07d4712-9da6-11da-831a-0006296cdc99
secHasPolicy: false
secDomainId: Default%testuser1
secDN: cn=testuser1,o=ibm,c=us
secPwdLastChanged: 20151208214816.0Z
secPwdValid: true
```

This is a summarized, hierarchical view of the output which shows some key object class definitions, attributes, and ACL placement:



```
dn: c=us,o=ibm
aclentry: ...ISAM ACLs...
 dn: cn=testuser1
 objectclass: inetOrgPerson
 userpassword: testpassword

dn: secAuthority=Default
objectclass: secAuthInfo
aclentry:...ISAM ACLs...
 dn: cn=Users
  dn: principalName=testuser1
  objectclass: secUser
  secDomainId: Default%testuser1
  secPwdLastChanged: 20151208214816.0Z
  secUUID: 9077caee-9da1-11da-8be4-0006296cdc99
```

Migrating to the minimal user data format

The Tivoli Directory Server cannot be migrated in place from the standard format to the minimal format.

To migrate to the minimal format, the Tivoli Directory Server must first be dumped to an LDIF format using the **db2ldif** command.

The LDIF data must be processed using the Security Access Manager **amldif2v6** command to produce a new LDIF file containing the migrated minimal format definition of the directory.

The previous Tivoli Directory Server database must be backed up, deleted, and rebuilt using the new, minimal format LDIF file.

You can find technical support for the **amldif2v6** tool at the IBM Security Access Manager website:

http://www.ibm.com/software/tivoli/products/access-mgr-e-bus

Tivoli Directory Server DB2 database

The Tivoli Directory Server stores a representation of directory in a DB2 database. The following information is an incomplete summary of the database tables used to implement the directory server and commands that can be used to display the database tables.

Although it is not necessary to access the Tivoli Directory Server with DB2 commands, this information might be useful to database administrators wanting more information. Tivoli Directory Server users must not depend upon the information in this section. It could change at any time.

The Tivoli Directory Server tables can be grouped into the following categories:

- LDAP entry table
- Subtree tables
- Attribute tables
- ACL tables
- Replication tables

The following sections describe more information about each of these categories and example DB2 commands that can be used against them.

For all the commands in this section, the name **ldapdb2** is used as the database instance owner account and the database name. Substitute this name as appropriate for the database configuration in which the commands are run.

Before executing the commands in this section, switch context to the DB2 instance owner account, for example by logging in to the DB2 instance owner account or using the following command (AIX, Linux, or Solaris operating system):

```
su – ldapdb2
```

On a Windows system use the following commands:

```
db2cmd
```

```
set DB2INSTANCE=ldapdb2
```

After switching the context, connect to the database using the following command:

```
db2 connect to ldapdb2
```

LDAP entry table

The LDAP entry category consists of a single table, the LDAP_ENTRY table.

This table holds the original LDIF definition of each LDAP entry. One of the columns is the Entry ID or EID column. The EID is used in all other tables of the database to identify the LDAP entry being referenced from the LDAP_ENTRY table. One of the uses of the LDAP_ENTRY table is to obtain the requested attribute values to be retrieved on a **ldapsearch** command.

Another use of the LDAP_ENTRY table is to evaluate the one level scope on a **ldapsearch** command. The one level scope is evaluated using the Parent EID or PEID column of the LDAP_ENTRY table.

To make it possible to include indexes on the Distinguished Name or DN, the LDAP_ENTRY table includes a DN_TRUNC column as well as a full nonsearchable DN column.

The following command describes the LDAP_ENTRY table. The show detail arguments are optional.

```
db2 describe table ldap_entry show detail
```

The following command finds the EID of a particular DN. The dn_trunc value must be in uppercase.

```
db2 "select eid from ldap_entry where dn_trunc = 'CN=TESTUSER1,O=IBM,C=US'
```

This command finds the DN of a particular EID:

```
db2 "select dn_trunc from ldap_entry where eid = 100"
```

This command finds the LDIF definition of a particular DN:

```
db2 "select ENTRYDATA from dap_entry where dn_trunc = 'CN=TESTUSER1,O=IBM,C=US'"
```

This command finds the DNs for the first 10 rows in the LDAP_ENTRY table:

```
db2 "select dn_trunc from ldap_entry fetch first 10 rows only"
```

This command finds the DNs for the next 10 rows in the LDAP_ENTRY table:

```
db2 "select dn_trunc from ldap_entry where eid > 10 fetch first 10 rows only"
```

This command finds all LDAP suffixes:

```
db2 "select dn_trunc from ldap_entry where peid = -1"
```

This command finds the DNs of all the immediate child entries (one level search) of the LDAP entry with DN "SECAUTHORITY=DEFAULT":

```
db2 "select dn_trunc from ldap_entry where peid in \
(select eid from ldap_entry where dn_trunc = 'SECAUTHORITY=DEFAULT')"
```

Subtree table

The subtree category also consists of the LDAP_DESC table and the LDAP_GRP_DESC table.

The LDAP_DESC table is used to evaluate the subtree scope on a **ldapsearch** command. This table contains a list of parent and child LDAP entry relationships using two columns:

- A Descendant EID or DEID column.
- An Ancestor EID or AEID column.

For each LDAP entry, there is a full list of parents for that LDAP entry in the LDAP_DESC table.Parent in this case includes immediate parent an all ancestors.

For example, the following command lists all the parents or ancestors of EID 100:

```
db2 "select * from ldap_desc where deid = 100"
```

Here is an example of a possible output:

```
DEIDAEID
----------- -----------
10011
10017
10023
10024
100100
```

This indicates that EID has four levels deep in the directory information tree.

The following command and its output are an example of such an EID:

```
db2 "select dn_trunc from ldap_entry where eid = 100"
DN_TRUNC
---------------------
PRINCIPAL=TESTUSER1,CN=USERS,SECAUTHORITY=DOM1,
CN=SUBDOMAINS,SECAUTHORITY=DEFAULT
```

The parents in this example are:

```
PRINCIPAL=TESTUSER1,CN=USERS,SECAUTHORITY=DOM1,CN=SUBDOMAINS,
SECAUTHORITY=DEFAULT
CN=USERS,SECAUTHORITY=DOM1,CN=SUBDOMAINS,SECAUTHORITY=DEFAULT
SECAUTHORITY=DOM1,CN=SUBDOMAINS,SECAUTHORITY=DEFAULT
CN=SUBDOMAINS,SECAUTHORITY=DEFAULT
SECAUTHORITY=DEFAULT
```

This command shows how DB2 tables can be joined into a single command.

It shows a command that lists all the parents of the LDAP entry with DN of:

```
principalname=testuser1,cn=users,secauthority=dom1,cn=subdomains, secauthorith=default
db2 "select * from ldap_desc where aeid in \
  (select eid from ldap_entry where dn_trunc = \
  'PRINCIPAL=TESTUSER1,CN=USERS,SECAUTHORITY=DOM1,CN=SUBDOMAINS,   \
SECAUTHORITY=DEFAULT')"
```

The LDAP_DESC is also used in subtree searches. The command finds all the child LDAP entries (both immediate and all descendants) for the LDAP entry with a DN of CN=USERS,SECAUTHORITY=DEFAULT:

```
db2 "select * from ldap_desc where aeid in \
 (select eid from ldap_entry where dn_trunc = \
'CN=USERS,SECAUTHORITY=DEFAULT')"
```

An example of the output is as follows:

```
DEIDAEID
----------- -----------
1212
200004212
200004312
200004412
200005612
200005712
```

```
200005812
```

The LDAP_GRP_DESC table is used to track nested group relationships.

Attribute tables

The attribute tables consist of one table per attribute that is used in the directory server.

The purpose of the attribute tables is to improve the performance of resolving search filters on LDAP searches, particularly when the attribute is indexed. The attribute tables are named by the attributes they represent. For example, the DB2 table for the cn attribute is named "cn".

This command describes the "cn" table:

```
db2 describe table cn
```

The output is as follows:

```
ColumnTypeType
nameschemanameLengthScale Nulls
----------------------------- --------- ----------------- -------- ----- ------
EIDSYSIBMINTEGER40 No
CNSYSIBMVARCHAR2560 No
CN_TSYSIBMVARCHAR2400 No
RCN_TSYSIBMVARCHAR2400 No
```

The CN column contains the full name for the attribute. The columns with names ending in "T" are the truncated to 240 character attribute name used for searching.

The column with a name beginning with "R" is the attribute name in reverse. This column is used for searching for attributes specified with a trailing wild card.

ACL tables

The ACL tables consist of the SRC, ACLPROP, OWNPROP, ENTRYOWNER, ACLPERM, ACLINHERIT.

The SRC table identifies from which LDAP entry a particular LDAP entry obtains the source for, or inherits, its ACL and owner information.

The SRC table is also the attribute table for the aclsource and entryowner attributes.

The following command describes the SRC table:

```
db2 describe table src
```

The output is as follows:

```
ColumnTypeType
nameschemanameLengthScale Nulls
----------------------------- --------- ----------------- -------- ----- ------
EIDSYSIBMINTEGER40 Yes
ACLSRCSYSIBMINTEGER40 Yes
OWNSRCSYSIBMINTEGER40 Yes
ACLTYPESYSIBMINTEGER40 Yes
```

The ACLPROP and OWNPROP are the attribute tables for the aclpropagate and ownerpropagate attributes.

The ACLPERM table is the attribute table for the aclentry attribute.

The ACLINHERIT table is the attribute table for the ibm-filterAclEntry attribute.

Replication tables

The replication tables consist of the REPLSTATUS, REPLCHGnnnn, REPLERROR, and several others.

There is one REPLCHGnnnn table for each replication context where nnnn is EID of the entry that is the base of the replication context.

The REPCHGnnnn implements the replication change table.

The REPLSTATUS is a pointer into the REPLCHGnnnn table indicating the last operation that was replicated.

Monitoring LDAP performance

To monitor performance for Tivoli Directory Server and Sun Java System Server Directory, enter the **ldapsearch** command:

```
ldapsearch -h ldap_host  -s base -b cn=monitor "objectclass=*"
```

where *ldap_host* is the name of the LDAP host.

This command returns several statistics. For the Tivoli Directory Server, a useful statistic for monitoring performance is opsinitiated, which indicates the number of LDAP operations that were initiated after the server started.

The **ldapsearch** command accounts for three of these operations. The throughput for any interval is:

- The difference between opsinitiated at the start and end of that interval.
- Less three for the **ldapsearch**.
- Divided by the length of the interval.

The following formula illustrates this calculation:

```
throughput for interval = (opsinitiated(at stop time) - \
opsinitiated(at start time) - 3) / (stop_time  - start_time)
```

Concurrent updates

Optimize update performance by making updates concurrently, for example, with multiple concurrent update clients.

The amount of performance improvement is limited by the speed of the disk system on the master and in a replicated environment on the replica servers. In a replicated environment, an increased update rate on the master server might require to tune a number of replica threads. The replica thread tuning, or consumer connections, might be required during the master server configuration.

Loading large numbers of users

You can add users to Security Access Manager in several ways. The suggested method depends on the number of users that you must add.

The following table helps determine the preferred method of adding Security Access Manager users:

Table 6. Preferred Methods for Adding Users

Number of Users to Create	Preferred Method
Less than 10,000	Security Access Manager **pdadmin** utility
More than 10,000	Tivoli Directory Server **bulkload** uti

For the **pdadmin** utility syntax, see the *IBM Security Access Manager: Command Reference*.

During this process, Security Access Manager adds users to the Tivoli Directory Server server. In turn, the Tivoli Directory Server server adds the user to the DB2 database.

Alternatively, you can bypass Security Access Manager and the Tivoli Directory Server server by loading users directly into the DB2 database. You can use the **bulkload** utility that is supplied with Tivoli Directory Server server.

Similar tools exist on other directory servers, such as Microsoft Active Directory and Sun Java System Server Directory. Some scripts might be useful in bulk loading users on the directory servers, particularly the servers that generate LDIF output. However, some modifications are required. The Tivoli Directory Server server **ldapadd** and **ldif2db** utilities are slower alternatives to the **bulkload** utility.

Before you load users, back up all critical Security Access Manager files and the user information in the DB2 database. The backup and restore procedures are in “1. Backing up the Tivoli Directory Server”

Tivoli Directory Server bulkload utility

The Tivoli Directory Server provides the **bulkload** utility, which:

- Parses the LDIF.
- Creates DB2 load.
- Loads the data directly into the DB2 database.

The advantage of the **bulkload** utility as compared to the **pdadmin** or **ldapadd** commands is performance. Bulkload is 10 - 20 times faster than any other method. The **bulkload** utility is not the right choice for every situation.

The fastest way to load Security Access Manager users is to use the **bulkload** utility from Tivoli Directory Server and use the Security Access Manager minimal data format. With this combination, it is not necessary to take the following actions:

- Turn off ACL processing in the **bulkload** utility.
- Run the Security Access Manager**fixacls.sh** tuning guide script.

If the LDIF data to be loaded includes:

- Only Security Access Manager users, there are no ACL definitions to process.
- A few LDAP entries with ACL definitions, the fastest load time comes from leaving ACL processing turned on in the **bulkload** utility. It is turned on by default. An example of ACL definitions is suffix LDAP entries.

You must make the following choices when you use the **bulkload** utility to load users:

- Security Access Manager user format: standard or minimal See "Security_Access_ Manager_user_format."

For more information about the **bulkload** utility, see the *Tivoli Directory Server Installation and Configuration Guide*.

Security Access Manager user format

For optimum load and update performance use the Security Access Manager minimal data format.

The minimal data format also improves authentication performance when the per user buffer pool requirement is decreased enough to allow a majority of Security Access Manager users to be cached in buffer pools. For more information regarding buffer pool memory requirements, refer to "Turning the DB2 buffer pools". The minimal data format uses 2 LDAP entries per Security Access Manager user and the standard format uses 4 LDAP entries per Security Access Manager user.

Tivoli Directory Server version

Tivoli Directory Server includes many performance improvements to the **bulkload** utility. It includes a chunk size option (**-k**) for breaking up the load into multiple passes.

This is an example usage:

```
bulkload -i <LDIF input file> -b -I ldapdb2 -k 2000000 -L /freedisk/ldapimport
```

The **bulkload** utility includes an option to add users to an existing group (**-G**). The input LDIF file requirements are slightly different from loading new LDAP entries.

Here is an example usage of loading groups:

```
bulkload -i <LDIF input file> -b -I ldapdb2 -G -L /freedisk/ldapimport
```

See the administration guide for the Tivoli Directory Server for more information.

Disk space requirements

Refer to Chapter 1, "Resource usage".

Using the fixacls.sh script

The Security Access Manager fixacls.sh tuning script fixes up LDAP access control lists (ACLs) when ACL processing is turned off during the use of the Tivoli Directory Server **bulkload** utility.

This script replaces all previous, similarly named scripts, for example **fixacls_propagate_parents.once**. The requirement for this script can be avoided by loading Security Access Manager users in the new Security Access Manager minimal format. See "Security Access Manager user data format" for more information about the Security Access Manager data format.

With the minimal data format, few LDAP ACLs are used so there is no need to turn off ACL processing during bulkload. With the standard data format, each Security Access Manager user requires an LDAP ACL.

The **bulkload** utility in Tivoli Directory Server can load millions of Security Access Manager users in the standard format with ACL processing turned on, but performs about 1.8 times faster with ACL processing turned off. It is not necessary to turn off ACL processing if Security Access Manager users are loaded in the new minimal data format, due to low ACL usage of this format.

If the **db2ldif** command is used to dump users in LDIF format from a Tivoli Directory Server upon which the current or a previous version of the fixacls.sh script has been run, the LDIF output does not include the ACL definitions for the Security Access Manager users. This is corrected by rerunning the fixacls.sh script after using the LDIF data to load the users into a new directory server.

The following are some situations in which this occurs:

- The Tivoli Directory Server is migrated to a new release using the method of dumping the database with **db2ldif**, unconfiguring, uninstalling, reinstalling, and reloading the directory from the **db2ldif** output file.
- A Tivoli Directory Server is loaded with the output of a **db2ldif** output file that came from a directory server upon which the fixacls.sh was previously run.

One exception to this is the migration of a Tivoli Directory Server from the standard Security Access Manager format to the minimal format using the **amldif2v6** utility. The minimal data format does not require the ACL fix ups that the standard format does.

To run the fixacls.sh script, switch user context to the root user, for example by logging in to the root user account, and execute fixacls.sh without parameters. The script prompts for any necessary input.

Making Security Access Manager users and groups

Certain LDAP entries must exist for a user to be defined as a Security Access Manager user. The purpose of the Security Access Manager scripts described in this section is to take an existing LDIF definition of LDAP users and add Security Access Manager LDAP entries to each of them. The resulting LDIF is used to load the users into the Tivoli Directory Server, for example with the **bulkload** utility.

This section describes the use of three Security Access Manager bulk-loading scripts.

Consider these scripts as examples from which usable scripts are derived. You can download the Performance Tuning scripts from the IBM Support site: http://www.ibm.com/support. To make the scripts usable, you must customize them for the particular environment in which they are to be used.

1. mk_test_users_ldif.sh
2. add_am_to_testusers_ldif.sh

These scripts are designed to be piped together, so that any piece can easily be replaced. The scripts have little error checking and must be read thoroughly before being used. Modifications are required.

mk_test_users_ldif.sh

The mk_test_users_ldif.sh script generates sample LDAP users without Security Access Manager LDAP entries defined. This script could be replaced with one that gets users from another Tivoli Directory Server server or some other source. Users could come from a database, such as a database containing employees, customers, or both. The script could also be replaced by a **cat** of a file containing the LDIF definition of users.

You can use this script as an example of the information needed to define an LDAP user for testing purposes. This script has the following usage:

mk_test_users_ldif.sh *start_user_number* *end_user_number* *user_prefix_dn password*

The output from this script is directed to the standard output device and contains the LDIF definition of the range of users requested on input. The script must be modified to indicate the LDAP suffix under which the users are to be located in the directory tree.

add_am_to_testusers_ldif.sh

The add_am_to_testusers_ldif.sh script adds the Security Access Manager LDAP entries to the input set of users.

The input *set of users* is read from the standard input device. The original user LDIF definition, along with the added LDAP entries, is output to the standard output device. You can modify the script to exclude the original user LDIF definition, where it can be used to import existing users to Security Access Manager.

Note: This script has a dependency on either the **uuidgen** or **pduuidgen** utility, the latter which is included with Security Access Manager. You must modify this script for your directory tree structure. You must also set the following variables within the script:

- **sechaspolicy**
- **secpwdvalid** (change the setting to false)
- **secpwdlastchanged**
- **secacctvalid**

For more information about the use of **pduuidgen**, see the *IBM Security Access Manager: Installation Guide.*

The script must be modified to identify where the Security Access Manager principalname attribute is obtained. As written, the script gets the principalname attribute from the directory user uid attribute. An alternative is to get the principalname attribute from the cn attribute, assuming that all users have a cn attribute.

This script takes three arguments, as follows:

add_am_to_testusers_ldif.sh *domain* [new|old] [*acls*]

where *domain* is either Default or a subdomain name, new|old is an optional keyword that specifies the Security Access Manager data format, and *acls* is an optional keyword that causes ACLs to be generated. new specifies the minimal format and old specifies the standard format.

Adding many members to a group

You can use three Security Access Manager scripts to define a large group, or one group with many members.

Consider these scripts as samples from which usable scripts are derived. You can download the Performance Tuning scripts from the IBM Support site: http://www.ibm.com/support.

To make the scripts usable, you must customize them for the particular environment in which they are to be used.

1. mk_ldap_group_ldif_stdin.sh
2. add_am_to_groups_ldif.sh

These scripts are piped together so that any piece can easily be replaced. The scripts have little error-checking and must be read thoroughly before being used. Modifications are required.

mk_ldap_group_ldif_stdin.sh

The mk_ldap_group_ldif_stdin.sh script creates or adds members to a test group taken from users read from the standard input device. The group does not contain the Security Access Manager LDAP entries. It is a group as defined by the Tivoli Directory Server server.

The DN of the group to be created is an argument to the script. The groups and group membership could come from some other source. This script could be replaced with a **cat** of a file containing the LDIF definition of the directory group object.

The script could be used to create a test group with many members. The usage is as follows:

mk_ldap_group_ldif_stdin.sh {create|add} *dn_of_group*

where the first option is either create or add, and *dn_of_group* is the distinguished name (DN) of the group to be created.

The create option generates the LDIF to create a group. The group is created with the specified number of members. The add option adds the specified members to an already existing group.

The script directs the LDIF output of the group definition to the standard output device.

add_am_to_groups_ldif.sh

The add_am_to_groups_ldif.sh script adds the Security Access Manager LDAP entries to the input group or set of groups.

The input group or groups is read from the standard input device. The original group LDIF definition along with the added LDAP entries is output to the standard output device.

You can modify the script to exclude the original group LDIF definition. You can use it to import existing groups to Security Access Manager. The script accepts any LDIF data from the input stream but acts on only the LDIF that creates group objects.

All other LDIF data are passed unchanged through to the standard output device.

For example, LDIF data that adds members to an existing group or creates non-group objects is echoed to the standard output device without change.

This script might also need to be modified to identify where the Security Access Manager's cn attribute is obtained. The cn attribute is the Security Access Manager's definition of the group name. As written, the script assumes that directory groups are named by the cn attribute.

This script requires one argument, *domain*, which is the name of the domain in which the group belongs, either Default or a subdomain name. The script can take an optional argument that specifies the Security Access Manager data format. If set to *new*, the minimal data format is used. If set to *old*, the standard format is used.

The script can take an optional third argument, which is the keyword **acl**. If specified, the script generates ACL entries for the added objects.

Using the group scripts together

You can use the example scripts to generate the LDIF output for a group that contains 10,000 users:

```
mk_test_users_ldif.sh 1 10000 cn=testusers,o=ibm,c=us test1pass |
mk_ldap_group_ldif_stdin.sh create cn=testgroup,o=ibm,c=us
```

```
> users.ldif
```

You can also combine the example scripts to generate the LDIF output to add 10,000 more members to the group created in the first example:

```
mk_test_users_ldif.sh 10000 20000 cn=testusers,o=ibm,c=us test1pass |
mk_ldap_group_ldif_stdin.sh add cn=testgroup,o=ibm,c=us
```

```
> group.ldif
```

The invocation of add_am_to_groups_ldif.sh is not necessary in the previous example, so you can replace it with the following:

```
mk_test_users_ldif.sh 1000020000 cn=testusers,o=ibm,c=us test1pass |
mk_ldap_group_ldif_stdin.sh add cn=testgroup,o=ibm,c=us > group.ldif
```

Limiting the pdadmin user list

Three conditions affect the maximum number of users that the **pdadmin user list** command can return:

- The *pattern max-return* argument for the **pdadmin user list** command:
  pdadmin> user list *pattern max-return*

  For example: pdadmin>user list *luca* 2 lists only two users.
- The max-search-size stanza entry in the [ldap] stanza of the ldap.conf configuration file.

  To indicate that there is no limit, set the maximum search size to 0. For example:

  max-search-size = 0
- The ibm-slapdSizeLimit parameter in the Tivoli Directory Server server slapd32.conf or ibmslapd.conf configuration file.

  To indicate that there is no limit, set the size limit to 0. For example:

  ibm-slapdSizeLimit = 0

  Note: This parameter affects all LDAP searches.

The output from the **pdadmin user list** command is restricted by the lesser of these three values.

LDAP cache

This section contains performance tuning tasks that are not typically used. However, there are some environments where these performance tuning tasks are useful.

The LDAP cache is more efficient in memory usage and faster than the DB2 cache, yet not as efficient as the Security Access Manager credential cache. Disadvantages to the LDAP cache are that it becomes invalidated on most update operations and can take a long time to load. The environments that benefit the most from LDAP caching are those with small registry sizes and few updates.

Keep in mind that increasing the LDAP cache size can cause the **slapd** or the **ibmslapd** process memory size to exceed system limits. For information about increasing these limits, see Chapter 6, “Tuning process size limits” of this document.

Setting the cache parameters

The LDAP cache parameters that are typically used with Security Access Manager are **RDBM_CACHE_SIZE** and **RDBM_FCACHE_SIZE**. These parameters are defined to LDAP with environment variables.

- To define the LDAP cache environment variables in the command shell (before starting the **slapd** or **ibmslapd** process), enter:

  stop LDAP

```
# (slapd or ibmslapd process)
export  RDBM_CACHE_SIZE=value
export  RDBM_FCACHE_SIZE=value
start LDAP
# (slapd or ibmslapd process)
```

- To define the LDAP cache environment variables in the slap32.conf or ibmslapd.conf configuration file, add the following entries to the file:

```
dn: cn=Front End,cn=Configuration
objectclass: top
objectclass: ibm-slapdFrontEnd
ibm-slapdSetEnv:  RDBM_CACHE_SIZE=value
ibm-slapdSetEnv:  RDBM_FCACHE_SIZE=value
```

For information about the definition of these and other LDAP cache parameters, see the Tivoli Directory Server documentation.

Choosing cache values

The primary use of the LDAP cache is for caching authenticated users.

Ways to choose values for the LDAP cache parameters include the following:

- Base the choice on the number of users to be cached.
- Base the choice on the amount of memory available for caching.

Note: Both ways require information about the memory usage per cached user and the number of cache entries used per cached user.

For Security Access Manager, the memory usage per cached user is approximately 3 KB and the number of cache entries used (per cached user) is four for the entry cache and five for the filter cache. The entry cache is controlled by RDBM_CACHE_SIZE and the filter cache is controlled by RDBM_FCACHE_SIZE. These approximations vary greatly with the various Security Access Manager configuration settings and usage.

The following conditions affect Security Access Manager use of LDAP cache resources:

- User-and-group-in-same-suffix setting in the webseald.conf and pdwebpi.conf files
- Number of LDAP suffixes in the slapd32.conf or ibmslapd.conf file

Choose cache values based on either the number of users you plan to create or on the amount of memory available.

- To choose the LDAP cache settings based on the number of users created, use the following formulas:

  LDAP entry cache size = (number of AM users) * 4
  LDAP filter cache size = (number of AM users) * 5
  Memory requirements = (number of AM users) * 3KB

- To choose the LDAP cache settings based on the amount of memory available, use the following formulas:

  number of AM users cached = desired memory usage / 3KB

  LDAP cache size = (number of AM users) * 4

  LDAP filter cache size = (number of AM users) * 5

The following table provides guidelines for cache size settings. Because these settings might not apply in every case, make certain to verify them (as explained in “Verifying the LAP cache resources usage”).

Table 7. Cache Size Settings

Number of Security Access Manager Users	Entry Cache Size	Filter Cache Size	Memory Usage	Additional Data Segments Needed (AIX)
10,000	40000	50000	30 MB	0
50,000	200000	250000	150 MB	0
100,000	400000	500000	300 B	1

Verifying the LDAP cache resources usage

Typically, there are two things to verify regarding the LDAP cache settings: one is whether cache misses were eliminated and another is if the LDAP process memory usage is as expected.

To verify that cache misses were eliminated, periodically enter on one line the following command while LDAP searches are in progress:

ldapsearch –h *ldap_host* -s base –b ’cn=monitor’ ’objectclass=*’ | grep *entry_cache_miss*

If all results come from the LDAP cache, the *entry_cache_miss* count remains constant throughout the usage of LDAP.

To verify that the LDAP cache memory usage is as expected, monitor the process memory growth as users are cached. Use the UNIX **ps** utility. For example, the following **ps** utility shows the current memory size of the LDAP process:

```
ps –e –o vsz –o –comm | grep slapd
```

Or:

```
ps –e –o vsz –o –comm | grep ibmslapd
```

Repeat this command periodically to determine the memory size of the slapd or the ibmslapd process when it levels off.

To verify that the LDAP cache memory usage does not exceed process memory size limits, watch the slapd or the ibmslapd process and verify that it does not end unexpectedly. If the slapd or ibmslapd process ends after increasing the LDAP cache, it is probably because it has exceeded the memory limits.

DB2 statement monitoring

The do_statement_monitoring.sh script can be used to assist in monitoring a DB2 statement.

Switch to the directory server DB2 instance owner, typically the **ldapdb2** user. Ensure that the script file ownership is that of the DB2 instance owner with the same group information as that user. Run the script without arguments.

The script prompts when it is ready for a test to be performed. Do not press Ctrl + C while running the script, or the statement monitor continues to run. Instead, press enter and wait for the script to return.

The proc_stmt_mon_output.awk script summarizes the output from the statement monitor. Run it as follows:

cat *dstate.out* | awk -f proc_stmt_mon_output.awk

where *dstate.out* is the output file from the do_statement_monitoring.sh script file.

The do_explain_sql.sh script calls the DB2 explain utility, which produces a report showing the query plan that the DB2 optimizer uses to process a SQL query.

Slow SQL queries can be input to this script to help identify why they are slow. Run the script without any arguments to get information about the input to the script.

Suffix search scope

Security Access Manager 7.0 has provided the user-and-group-in-same-suffix tuning parameter to help limit the number of LDAP searches that must be performed.

Security Access Manager 7.0 also uses the ibm-allgroups parameter for helping to limit search scope. This means that the user_and_group_in_same_suffix parameter is now only applicable under two conditions:

1. When Sun or Novell LDAP is used.
2. When dynamic-groups-enabled = no in the **ldap.conf** file.

Suffix search order

Security Access Manager dynamically adjusts the search ordering of the LDAP suffixes used to locate users in the LDAP directory tree during authentication.

The search order algorithm attempts to minimize the number of LDAP operations required to find the suffix under which the user object resides. You can reduce the number of suffix search operations and improve performance by minimizing the number of suffixes that exist in the directory server.

Although you are not supposed to typically modify the LDAP suffix search order, there are tuning options applicable to any LDAP directory server supported by Security Access Manager, for example the Tivoli Directory Server.

Ignoring suffixes

Security Access Manager has tuning options to ignore one or more of the LDAP suffixes. For instance, you can configure those suffixes that do not contain Security Access Manager users or groups and must not be searched. This is controlled by the value of the ignore-suffix stanza entry in the ldap.conf configuration file on the IBM Security Access Manager appliance.

By default, the ldap.conf file contains an ignore-suffix stanza entry setting to ignore the following suffix: cn=ibmpolicies.

In terms of performance, ignoring suffixes is better than using LDAP ACLs to exclude certain suffixes from usage by Security Access Manager. For example, it takes fewer search operations to determine that a user does not exist if the suffixes that do not contain Security Access Manager users are ignored.

Another approach is to move all non Security Access Manager objects to a subtree under a common suffix and protect that subtree with an LDAP ACL. Fewer suffixes result in fewer LDAP search operations.

Security Access Manager always unconditionally ignores the following LDAP suffixes:

cn=localhost
cn=pwdpolicy
cn=configuration

Ordering of the suffix list

Using an ordered list of suffixes, Security Access Manager finds a user by searching each suffix in the list, in order, and stopping when a match is found.

The ordering of the suffix list is periodically adjusted through incrementing suffix counters when a user is found. When any suffix counter reaches a trigger value, which defaults to 10, the suffix search list is adjusted and all counters are reset to 0.

The default adjustment uses a most frequently used approach in which the suffixes are sorted from highest to lowest suffix counter value.

The following environment variables are provided to control the search ordering algorithm:

PD_SUFFIX_TRIGGER_COUNT

- Default: 10
- Acceptable values: a positive integer

This environment variable defines the suffix counter trigger value and controls how often the search list is adjusted. When any suffix counter reaches this value, the suffix search list is adjusted and all counters are reset to zero.

PD_SUFFIX_MAX_ITERATIONS

- Default: -1 (which means unlimited iterations)
- Acceptable values: -1, 0, or a positive integer

This environment variable defines the maximum number of consecutive executions of the adjustment algorithm that results in no change to a suffix search list. The adjustment algorithm is executed each time a suffix counter reaches the trigger value and attempts to reorder the suffix search list. Any execution of the adjustment algorithm that changes the suffix search list resets the iteration count to zero. A value of -1 indicates no limit to the number of adjustments. A value of 0 indicates no adjustments. With no adjustments, the order of the suffix search list is statically initialized to the order in which the LDAP server returns the set of suffixes it maintains.

For values other than -1 and 0, when the iteration count reaches that value, the suffix search list becomes static, with the assumption that the suffix search order is correct. Each time the Security Access Manager server is started, the iteration count is reset, allowing adjustments to again occur until the maximum iteration count is reached.

PD_SUFFIX_ORDERING_BUBBLE

- Default: most frequently used adjustment algorithm
- Acceptable values: any, must simply be defined

If defined, this environment variable replaces the most frequently used default adjustment algorithm with one that places the suffix with the highest counter value at the front of the list, leaving the remaining order of the list unchanged.

Tuning the network keep alive frequency for firewalls

When the LDAP server is protected behind a firewall, socket connections might time out, resulting in intermittent authentication failures.

The socket connection failures are due to a mismatch between the firewall connection timeout setting and the operating system frequency of sending keep alive network packets to keep the connection alive. If socket connection failures occur, decrease the operating system network parameters that control the time between sending keep alive packets, sometimes called the keep alive interval.

The name of the parameters that control keep alive frequency vary with each operating system. Set the keep alive interval to be less than the firewall connection timeout. If unsure of the firewall setting, try 2 minutes.

The following information describes setting this parameter for the AIX operating system:

Use the command **no** to set attributes and values for performance tuning:

no -o *attributename=value*

tcp_keepidle

The tcp_keepidle parameter specifies the interval, measured in half seconds, of inactivity that causes TCP to generate a KEEPALIVE transmission for an application that requests them.

tcp_keepidle defaults to 14400 (two hours). This is the parameter that controls the keep alive interval for AIX.

Reduce tcp_keepidle to be less than the firewall connection timeout. If unsure, set tcp_keepidle to 240 (2 minutes).

tcp_keepintvl

The tcp_keepintvl parameter specifies the interval, measured in half seconds, between the nine retries that are attempted if a KEEPALIVE transmission is not acknowledged.

tcp_keepidle defaults to 150 (75 seconds). This parameter is related to the keep alive interval, but does not typically need to be modified.

For operating systems other than AIX, refer to the operating system documentation for information about setting the network keep alive interval.

Chapter 4. Tuning other directory servers

You can tune the performance of directory servers, other than the Tivoli Directory Server that are supported.

The following tuning topics from the Tivoli Directory Server tuning sections also might be applicable to other directory servers that are supported by Security Access Manager:

- Security Access Manager user data format
- Monitoring LDAP performance
- Making Security Access Manager users and groups
- Tuning the network keep alive frequently for firewalls

One significant difference between directory servers is how access control lists (ACLs) are supported. The following sections describe only ACLs that are as supported by the Tivoli Directory Server.

Attribute indexes

You can index attributes to improve directory performance.

You must index certain attributes to achieve good performance on directories with large numbers of users. For example, directories with more than a thousand users typically require indexing.

Security Access Manager automatically indexes attributes that are important to Tivoli Directory Server and Microsoft Active Directory performance.

For other directory servers, you must manually create indexes by using the utilities that come with the directory server. The Novell eDirectory and Sun Java System Directory Server are examples of directories that require manual indexing.

Index the following attributes for optimum authentication performance:

```
principalName
secDN
secDomainId
secCERTDN
```

Index the following attributes, in addition to the ones in the previous example, for optimum administrative performance:

```
secAuthority
secUUID
sys
```

Index the uid attribute if the directory server users are shared with WebSphere:

Tuning Sun Java System Directory

Customize the Sun Java System Directory *look-through limit* parameter to avoid the LDAP_ADMINLIMIT_EXCEEDED error.

About this task

Note: The Sun Java System Directory was formerly called iPlanet Directory Server.

Installing Security Access Manager on the Sun Java System Directory, version 5, registry can cause an LDAP_ADMINLIMIT_EXCEEDED status. This status indicates that the user registry contains more entries than the limit in the *look-through limit* parameter. Security Access Manager treats this status as an error.

You can change the limits in the *look-through limit* parameter.

Procedure

1. Access the Sun Java System Directory Console.
2. Select the **Configuration** tab.
3. Expand the **Data** entry.
4. Select the **Database Settings**.
5. Select the **LDBM plug-in Settings** tab.
6. In the **Look-through Limit** field, enter the maximum number of entries you want the server to check in response to a search request.
   - The default *look-through limit* is 5000.
   - If you do not want to set a limit, enter **-1**.

Chapter 5. Tuning Security Access Manager Components

This chapter provides tuning information for Security Access Manager components such as reverse proxy and the authorization server.

Configuration file tuning

Modifying the Configuration files

The tuning parameters in server configuration files must be modified using the Access Manager Local Manager Interface.

Modifying reverse proxy configuration file

Perform the following steps to modify the reverse proxy configuration file:

1. Logon to LMI using the administrator username and password. A typical id of an administrative user in most installations is 'admin'
2. Select Secure Web Settings → Reverse Proxy.
3. Select the Reverse proxy instance
4. select Manage – >Configuration –> Edit Configuration.
5. Make the necessary changes and save you changes.
6. Deploy the changes.
7. Restart the reverse proxy instance to make the changes effective.

Modifying the authorization configuration file

Perform the following steps to modify the reverse proxy configuration file.

1. Logon to LMI using the administrator username and password. A typical id of an administrative user in most installations is 'admin'
2. Select Secure Web Settings → Reverse Proxy.
3. Select the Reverse proxy instance
4. select Manage – Configuration – Edit Configuration.
5. Make the necessary changes and save you changes.
6. Deploy the changes.
7. Restart the reverse proxy instance to make the changes effective.

Modifying the LDAP Registry configuration file

1. Logon to LMI using the administrator username and password. A typical id of an administrative user in most installations is 'admin'
2. Select Secure Web Settings → Runtime Component

3. select Manage – Configuration – Edit Configuration.
4. Select ldap.conf
5. Make the necessary changes and save you changes.
6. Deploy the changes.

Road map for configuration file tuning

The stanzas in this section are found in the configuration files for reverse proxy and authorization servers.

Common to reverse proxy and authorization server

[ldap] stanza

- "auth-using-compare"
- "user-and-group-in-same-suffix"
- "LDAP root administrator account (cn=root)"
- "Enabling the LDAP cache for Java authentication"
- "client"
- "client-asynch-auth-binds"
- "IRA cache tuning information"

[aznapi-configuration] stanza

- "policy-cache-size"

reverse proxy only

[ssl] and [session] stanzas

- "SSL session cache, user credential cache, and memory use"

[server] stanza

- "WebSEAL worker threads"

auth-using-compare

The auth-using-compare stanza entry uses password comparison to enable or disable authentication. When disabled, authentication is performed with the LDAP bind.

Use the default setting of yes for the auth-using-compare stanza entry in the following server configuration files:

- Reverse proxy
- Authorization server

The auth-using-compare stanza entry affects only the performance of the Tivoli Directory Server server and varies with the size of the directory. This stanza entry has little or no effect on the processor requirements of the reverse proxy server.

As compared with a setting of yes, The authentication throughput of the Tivoli Directory Server server with the no setting is from 5% faster with 10 thousand users in the directory to 30% slower with 1 million users in the directory. There is evidence that the slower performance is due to lock contention, so the effect might also vary with the number of processors. These results are based on a 4-CPU system.

With the auth-using-compare stanza entry set to no, Security Access Manager authenticates using the traditional LDAP bind and unbind commands.

With the auth-using-compare stanza entry set to yes, Security Access Manager authenticates with the IBM LDAP unique search and compare command.

The auth-using-compare stanza entry is ignored when the Sun Java System Directory Server is used.

User-and-group-in-same-suffix

The user-and-group-in-same-suffix stanza entry indicates whether the groups, in which a user is a member, are defined in the same LDAP suffix as the user definition

A default setting for the user-and-group-in-same-suffix stanza entry in the following server configuration files is no:

- Reverse proxy
- Authorization Server

Setting this stanza entry to yes might help the performance of some supported directory servers, but has no impact on the Tivoli Directory Server.

The following information is for those directory servers that are impacted by this stanza entry:

- When this value is set to yes, Security Access Manager assumes the user, and the group or groups that the user is a member of, are in the same suffix.
- When this value is set to no, Security Access Manager searches every suffix for a given users group membership.
- Setting the user-and-group-in-same-suffix stanza entry to yes reduces LDAP searches for authentication.

  Authentication performance is directly related to the number of LDAP search operations that are performed.

With Tivoli Directory Server, Security Access Manager searches for group membership on all suffixes in one command by using the ibm-allgroups attribute.

LDAP root administrator account (cn=root)

When Security Access Manager is configured, it creates LDAP user accounts that are used to access the LDAP directory. The Tivoli Directory Server server administrator can set ACLs in the directory that allow or deny Security Access Manager server users access to parts of the directory tree.

For the Tivoli Directory Server server, the additional ACL checking associated with each LDAP search results in a slight performance reduction of approximately 10 percent.

The Tivoli Directory Server server does not check the ACL when the LDAP root administrator account is used to access the directory. By changing the Security Access Manager configuration to use the LDAP root administrators account (typically cn=root), ACL checking is eliminated.

The stanza entries that control this are bind-dn and the bind-pwd stanza entries in the Security Access Manager server configuration files. In Security Access Manager, the bind-pwd stanza entry is obfuscated. Refer the server configuration file information in the *IBM Security Access Manager: Administration Guide* for information about changing values for these stanza entries.

Enabling the LDAP cache for Java authentication

When a Java application uses the Security Access Manager authorization Java class PDPrincipal object to construct a user credential, a call is made to the remote

Security Access Manager authorization server. The Security Access Manager authorization server makes calls to the LDAP directory server as part of creating the credential.

The performance of instantiating the PDPrincipal object is slow unless the LDAP cache is enabled on the authorization server. By default, the LDAP cache is disabled on the authorization server. The parameter that enables the LDAP cache is located in the [ldap] stanza of the authorization configuration server file and is named **cache-enabled**.

The following example setting enables the cache:

```
[ldap]
cache-enabled = yes
```

Examples of applications that benefit from this change include:

- IBM Tivoli® Federated Identity Manager
- A customer-written external authorization interface (EAI) server that uses the authorization Java class PDPrincipal object to create a Privilege Attribute Certificate (PAC) for each user.

The Security Access Manager LDAP cache saves the results of LDAP queries to improve the performance of authentication. The LDAP cache is also known as the IRA cache.

For reverse proxy the LDAP cache is enabled by default.

For the authorization server the LDAP cache is disabled by default.

The LDAP cache setting must be changed from its default of disabled to enabled for the authorization server.

The expire time is controlled by the following stanza entries in the [ldap] stanza:

```
[ldap]
cache-user-expire-time
cache-group-expire-time
cache-policy-expire-time
```

Refer to the *IBM Security Access Manager: Administration Guide* for more information about these parameters.

client-async-auth-binds

When a reverse proxy is authenticating to Active Directory, by default it will only use a single connection, so it will authenticate one user at a time. You can improve login performance by telling the reverse proxy to authenticate multiple users concurrently. This is done by setting client-async-auth-binds = yes in the [ldap] stanza. If the client-async-auth-binds stanza entry is not already present, it should be added to the [ldap] stanza of the applicable reverse proxy configuration file.

```
[ldap]
client-async-auth-binds = yes
```

IRA cache tuning information

This section explains how to configure the IRA cache for Tivoli Directory Server and Sun Java System Directory.

The Security Access Manager IRA cache is a cache of LDAP information; therefore, the IRA cache applies to only Security Access Manager configurations that use LDAP as the registry interface. The following are some of the directory products that use LDAP as the registry interface:

- Tivoli Directory Server
- Sun Java System Directory

Authentication (user login) performance is fastest when authentication information comes from the Security Access Manager IRA cache. The disadvantage to the IRA cache is that the information can become stale and not accurately reflect recent updates in the Tivoli Directory Server server. For example, an update to a users group membership is not reflected in the IRA cache until that users cache entry times out. Another potential disadvantage is that the information in the cache might be considered sensitive. In many environments this information is not considered sensitive. In no case is the user password cached.

The following information is stored in this IRA cache:

- UUID
- principalName
- DN
- accountValid
- passwordValid
- group membership
- user-specific policy (such as when password was last changed)

The IRA cache can be tuned by setting the the parameters that are prefixed with “cache” in the [ldap] stanzas of the Security Access Manager Configuration files.

The following cache information can be disabled by setting the appropriate parameters in the configuration files.

- accountValid

- passwordValid
- group membership

This information is controlled with the following IRA configuration parameters:

- CACHE_CACHE_USER_VALID_FLAGS
- CACHE_CACHE_GROUP_MEMBERSHIP

Although not recommended, the cache can be disabled through the cache-enabled

stanza entry in the [ldap] stanza of the reverse proxy Server configuration file.

Do not completely disable the IRA cache. The IRA cache provides a large performance improvement in authenticating even non-cached users because of the reduction in redundant registry lookups that the cache provides. By leaving the user cache timeout at its default of 30 seconds, the stale cache and sensitive information concerns are addressed.

As long as the time-outs are left short, it is also recommended that the cache be enabled to use the **accountValid**, **passwordValid**, and **group membership** information. The default value for each of these is enable.

If you use the IRA cache to improve authentication performance, the **size** and **timeout** values of the cache must be increased. Also, you need to decide whether to disable the use of the **accountValid**, **passwordValid**, and **group membership** information. By disabling the use of that information, the number of registry lookups increases and removes some of the performance benefit of the IRA cache.

Do not increase the IRA cache if you expect more than 10,000 users to authenticate. The caches are searched sequentially. Caches are not intended for large volumes of data. They are intended for shorter lived data.

Increasing the IRA cache parameters can cause reverse proxy process memory usage to exceed system limits. See “Chapter 6, “Tuning process size limits” for information about increasing these limits.

User cache

CACHE_USER_CACHE_SIZE

Default: 256 entries

Recommended: If the IRA cache is to be used to improve authentication performance, set this to the number of users that are expected to authenticate, plus some overhead (for example, 15%) for peak usage but not greater than 10,000. Otherwise, leave it the default value.

Controls the number of user entries cached in memory for the reverse proxy LDAP client cache. The user entry contains:

UUID
principalName
DN
accountValid
passwordValid

group membership
user specific policy
...

CACHE_USER_EXPIRE_TIME

Default: 30 seconds

Recommended: If the IRA cache is to be used to improve authentication performance, set this high (for example, 28800 seconds for 8 hours, 86400 for daily or 604800 for weekly). Otherwise, leave it the default value.

Controls the amount of time in seconds that the user entry is cached in memory for the reverse proxy LDAP client cache. After the time expires, reverse proxy must reissue the LDAP queries to refresh the reverse proxy LDAP client cache.

Each user request is 2 LDAP queries

CACHE_CACHE_USER_VALID_FLAGS

Default: 1 (Yes)

Recommended: If the IRA cache is to be used to improve authentication performance, consider setting this to 0 (No). Otherwise, leave it the default value.

This flag is checked only when the user logs in.

If the user entry exists in the reverse proxy LDAP client cache and the current setting is:

- 1 (Yes)

  The **accountValid** and **passwordValid** are obtained from the user entry in cache, even though the **accountValid** and **passwordValid** might have changed on the Tivoli Directory Server server. Updated **accountValid** and **passwordValid** changes might not be visible to reverse proxy until after the user entry in the reverse proxy LDAP client cache has expired (up to IRA_USER_EXPIRE_TIME seconds) and then the user logs in.This is beneficial for stress testing when the same user logs in and logs off repeatedly; however, the administrator **accountValid** and **passwordValid** changes might appear to be random in production.

- 0 (No)

  Each time the user logs in the **accountValid** and **passwordValid** flags from the user entry in the reverse proxy LDAP client cache are bypassed to determine the current settings regardless of the age of the user entry in the cache.

  Therefore, each time the user logs in, an LDAP query is issued to the Tivoli Directory Server server.

  If an administrator changes the **passwordValid**or **accountValid** flags, then reverse proxy can retrieve the current settings after the user logs in the next time.

  This is beneficial for production due to the predictable behavior; however, during stress tests when the same user logs in and logs off repeatedly, this action could create additional overhead bypassing the reverse proxy LDAP client cache for each login.

CACHE_CACHE_GROUP_MEMBERSHIP

Default: 1 (Yes)

Recommended: If the IRA cache is to be used to improve authentication performance, consider setting this to 0 (No). Otherwise, leave it the default value.

This flag is only effective when the user logs in.

If the user entry exists in the reverse proxy LDAP client cache and current setting is:

- 1 (Yes)

  The user group memberships are obtained from the user entry in cache, even though the group memberships might have changed on the Tivoli Directory Server server. Any group membership additions or deletions are not visible to the reverse proxy until after the user entry in the reverse proxy LDAP client cache has expired (potentially up to CACHE_USER_EXPIRE_TIME) and then the user logs in. This is beneficial for stress testing when the same user logs in and logs off repeatedly; however, the administrator might not be able to perceive addition or deletion of a user group immediately.

- 0 (No)

  Each time the user logs in the user group membership from the user entry in the reverse proxy LDAP client cache is bypassed to determine to which groups, the user currently belongs.

  Therefore, each time the user logs in, an LDAP query is issued to the Tivoli Directory Server server.

  If an administrator adds or deletes a user group, the reverse proxy can see the current group membership after the user logs in the next time.

  This is beneficial for production due to the predictable behavior; however, during stress tests when the same user logs in and logs off repeatedly, it could create additional overhead bypassing the reverse proxy LDAP client cache for each login.

Group cache

CACHE_GROUP_CACHE_SIZE

Default: 64 entries

Recommended: Set to the total number of groups in which users have membership plus some overhead (for example, 15%) for peak usage but not greater than 10,000.

Controls the number of group entries cached in memory for the reverse proxy LDAP client cache. The group entry contains:

UUID
groupName DN
...

CACHE_GROUP_EXPIRE_TIME
Default: 300 seconds (5 minutes)

Recommended: 28800 seconds (8 hours)

Controls the amount of time in seconds that the group entry is cached in memory. After the time expires, the reverse proxy must reissue the LDAP queries to refresh the reverse proxy LDAP client cache. The user can belong to multiple groups. Each group for the user not in cache is 2 LDAP queries.

CACHE_GLOBAL_POLICY_EXPIRE_TIME

Default: 30 seconds

Recommended: 30 seconds

Controls the amount of time in seconds that the global policy change is effective.

Policy-cache-size

The policy-cache-size stanza entry dictates the maximum number of entries allowed for the in-memory policy cache.

The default value is 32768 entries, if no value is set.

The minimum value is 256 entries. The maximum value is 524288 entries.

The default setting of the policy-cache-size stanza entry might not be sufficient to provide the best performance. If the stanza entry is not already present, it should be added to the [aznapi-configuration] stanza of the applicable reverse proxy configuration file.

The following information provides some guidelines on how to set an appropriate value:

- The size of the in-memory policy cache index is configurable.

  The cache consists of the policy and the relationships between the policy and the resources. The knowledge that a resource has no directly associated policy is also cached.
- The size of the cache index must be relative to the number of policy objects that are defined and the number of resources that are protected.

  An internal calculation is performed to derive the cache size by using the policy database. This calculation might not be large enough because it does not take into account the resources that do not have a policy directly attached. Resources that do not have a policy attached might not be stored in the policy database.

A reasonable algorithm to begin with is:

(number of policy objects * 3) + (number of protected resources * 3)

A larger cache potentially improves the application performance, but uses additional memory as well.

The size is specified as the number of expected entries. The default size is 32768 with a minimum of 256 and a maximum of 524288. A size less than 256, becomes 256. A size greater than 524288 becomes 524288. A non-numeric size becomes 32768.

This is an example of setting the policy cache size:

```
[aznapi-configuration]
policy-cache-size = 32768
```

SSL session cache, user credential cache, and memory use

The following stanza entries in the reverse proxy Configuration file are relevant to these caches:

```
[ssl]
ssl-v2-timeout = 100
ssl-v3-timeout = 7200
ssl-max-entries = 4096

[session]
max-entries = 4096
timeout = 3600
inactive-timeout = 600
```

The ssl-max-entries and max-entries stanza entries control the size of the SSL and credential caches.

Increases to the SSL and credential cache sizes can cause reverse proxy process memory usage to exceed system limits. For information about increasing these limits, see Chapter 6, "Tuning process size limits".

The SSL session cache uses about 250 bytes of process memory per entry and the credential cache uses about 7.5KB of process memory per entry.

Note: The [ssl] configuration file stanza is not always available in the plug-in for Web Servers configuration file.

Reverse proxy worker threads

The worker-threads stanza entry in the reverse proxy configuration file specifies the number of worker threads that the reverse proxy allocates for handling incoming requests. In most cases, it is not necessary to adjust the worker-threads stanza entry.

When all worker threads become busy processing requests, thread starvation occurs for new requests and leads to request timeouts, failures, and poor performance. In such cases, increase the number of worker threads to address the problem. Following are situations which result in high worker thread usage:

- A slow back-end Web server can block a worker thread until that server responds.
- If all 100 of the default worker threads become blocked, thread starvation occurs and the reverse proxy is not able to process any further requests.
- Slow network connections between browsers and the reverse proxy server can result in many worker threads that wait for the completion of send or receive requests.

- A long running dynamic web page, such as a CGI script or Java Server Page (JSP), on the reverse proxy server can block a worker thread until the script is complete.

  The page must be moved instead of increasing worker threads. The reverse proxy does a fork function to process a dynamic Web page. The fork function makes a copy of the memory used by the parent process and creates a process. Increasing the number of worker threads also increases the reverse proxy process memory size.

  Increasing memory size slows down the fork function and negates any performance improvement from increasing the number of worker threads. When this situation occurs, move the dynamic Web page to a back-end Web server, instead of increasing the number of worker threads.

Detecting worker thread starvation

About this task

One way to detect worker threads starvation is to check the reverse proxy message log. When the soft or hard thread worker limit is reached, a message is generated in this message log. The soft and hard worker thread limits are controlled by the worker-thread-hard-limit and worker-thread-soft-limit stanza entries in the reverse proxy configuration file. The unit of measure for these limits is percentage of total threads. A value of 100 indicates that a message is generated when all worker threads are in use.

When the number of active worker threads reaches the *hard* limit, and the hard limit is not the maximum value of 100, in addition to generating a message, a 503, Service Unavailable error is returned to the requesting web browser.

The soft and hard limits can also be specified on a per-junction basis through the **pdadmin junction create** command with the **–l** and **–L** options. See the information about creating a junction for an initial server in the *IBM Security Access Manager: WebSEAL Administration Guide*.

Another way to detect a worker thread starvation situation is to use the reverse proxy active worker thread statistic. To get this information, follow these steps:

Procedure

1. Issue the following command:

   pdadmin -a *sec_master* -p *password* server list

   where *sec_master* is the security master account which defaults to sec_master and *password* is the security master account password. This command returns a list of the names of all the reverse proxy servers that are configured into the policy server.

2. Find the name of the reverse proxy server of interest, which was returned on the previous command. Then, issue the following command in one line against that web server to determine the number of active worker threads:

pdadmin -a *sec_master* -p *password* server task *reverse_proxy_server*

stats get pdweb.threads

where *reverse_proxy_server* is the name returned on the first command for the reverse proxy server of interest. This command returns the number of active threads and the number of total threads.

3. If the number of active and total threads is equal or near equal, increase the number of worker threads, and restart the reverse proxy server.

Results

Other indications that worker thread starvation is occurring includes:

- Web browsers get timeout errors.
- Response times are high, yet the reverse proxy server CPU utilization is low.
- Thread stack dumps indicate reverse proxy worker threads are blocked on receive and not on a mutex, meaning that they are waiting for work.
- TCP/IP packet traces indicate the number of active requests that the reverse proxy is processing is equal to the number of worker threads.

Adding worker threads versus adding instances

Reverse proxy worker threads scale well. Do not run multiple instances, unless you need more worker threads than one instance can handle.

Determining the number of worker threads to use

Processing concurrent requests at peak loads requires a sufficient number of worker threads to handle the expected number of requests. You can determine the number of expected concurrent requests with the following formula:

(expected rate of requests per second) * (expected response time in seconds per request)

The response time required for the calculation is the amount of time it takes the reverse proxy to write all the bytes to the client connection. Several factors can affect the response time, such as:

- Slow backend applications
- Clients that are slow to read a large response
- Slow network connections (for example dial-up clients)

You can reduce the impact of slow clients by increasing the buffering in the TCP layer. For example, on AIX you adjust TCP_sendspace.

After the worker thread writes all the data to the socket, it then handles the next request.

Worker thread resource usage

Each additional worker thread consumes about 450 KB of virtual storage and two TCP/IP sockets. Physical memory is limited by the sum of physical memory and the paging space on the system. To prevent thrashing, process memory usage must be limited to the available physical memory.

Determining Reverse proxy Worker thread limits

This section describes the maximum number of worker threads for the reverse proxy.

Reverse proxy was tested to a maximum of:

- 30000 worker threads on hardware and virtual appliances.

Note: 30000 worker threads is the total maximum of worker threads; in case you have a multiple reverse proxy instances it is shared resource between those instances. You must thoroughly test and evaluate the appropriateness of this setting for your environment.

This section provides formula for calculating the maximum number of available threads, depending on your environment. The formula use the following three variables, which you must determine before you attempt any calculation:

Max-sockets

Determine the maximum number of outgoing connections available on the appliance according to the following chart.

Table 8. Maximum number of sockets available for appliance

Operating System	**Max-sockets**
Appliance	28233 to 64510 **Note:** The default dynamic port range is 28233, but you can increase the range up to 64510. For more information, see Technote http://www-01.ibm.com/support/docview.wss?uid=swg21960611

Reserve-file-descriptors

Equal to the value of the reserve-file-descriptors parameter.

Note: Reserve-file-descriptors must be set to a minimum of 65. If it is configured it to a value less than 65, it is overridden to the value 65.

Junctioned Servers

Equal to the number of the connections used by threads monitoring the junctioned servers.

Examples for a number of junctioned servers:

- if you have a junction with two servers on it, then each one of those servers will use a connection
- if you have two junctions and each junction has two servers, than that will be four used connections.

Formula

Use the formula maxWorkerThreads = (max-sockets– reserve-file-descriptors– junctioned servers) / 2

to calculate maximum value for the number of worker threads.

Tuning I/O buffers

In most cases, you do not have to adjust the I/O buffer size that is used by the reverse proxy to communicate with clients or junctioned applications. However, you might see a benefit from increasing the client-facing I/O buffer size if the objects requested are mostly large.

The io-buffer-size in the [server] stanza of the reverse proxy configuration file can be increased from its default size of 8192. For SSL, the highest useful value is 16320. This value matches the largest size write that GSKit does.

Tuning Persistent Connections

Tune the persistent connection settings between reverse proxy and a junctioned application server to minimize the time and cost you spend setting up new connections.

Persistent connections between Reverse proxy and a junctioned application server

IBM Tivoli Access Manager version 6.1.1 and later supports persistent connections between the reverse proxy and a junctioned application server.

By enabling this feature, the reverse proxy caches a pool of open connections to each junctioned server for future use.

A new HTTP request results in the reverse proxy retrieving an open connection from the pool, if available. After the request is processed, the open connection is returned to the pool unless either of the following circumstances occur:

- The configured limit on the pool size is reached.
- The HTTP response from the junctioned server includes the connection:close header.

Benefits of persistent connections

The benefits of enabling persistent connections include:

- Faster response time for request
- Less CPU usage for both the reverse proxy and the junctioned application server

Therefore, persistent connections can significantly improve reverse proxy performance.

Note: If the junctioned server is configured to use short connection timeouts, the usefulness of the reverse proxy persistent connection pool is limited.

Security implications of persistent connections

When persistent connections from reverse proxy to a junctioned application server are enabled, a single connection can potentially be used for multiple users. This setup might not be appropriate for junctioned applications that tie a user identity to an open connection.

If SSL is enabled between reverse proxy and the junctioned application server, then using persistent connections means that the SSL handshake and establishment is not done for each request.

Multiple requests might reuse a single SSL connection from the reverse proxy to the junctioned application server.

Tuning persistent connection

You can tune the persistent connections between the reverse proxy and a junctioned application server for optimized processing.

About this task

The reverse proxy does not impose any upper limit for the size of the persistent connection pool cache. You can set the size of the cache to any number you want.

There is no point to setting it higher than the number of worker threads, as that is the maximum number of concurrent connections that the reverse proxy opens to a junctioned server.

Set the file descriptor reverse proxy server process limit high enough to accommodate the maximum number of open connections that you expect at any time because each open connection ties up a file descriptor.

The junctioned server for each junction has its own connection pool.

Procedure

1. Calculate the number of cached connections for each junction that requires persistent connections.

   The value must be the expected number of concurrent requests to each junctioned server for that junction.

   The **[worker-threads]** parameter is a reasonable upper limit for each junction. The parameter controls concurrent incoming requests that can be serviced by the reverse proxy instance.

   See the *worker-threads* section of the *[server] stanza* chapter in the IBM Security Access Manager Stanza Reference Guide for details.

2. Calculate the connection timeout value to be less than the maximum connection lifetime supported by the junctioned server.

   Sample setting

   ```
   [server]
   worker-threads = 300

   [junction]
   max-cached-persistent-connections = 50
   persistent-con-timeout = 5

   [junction:appserver1]
   max-cached-persistent-connections = 300

   [junction:appserver2]
   max-cached-persistent-connections = 250
   persistent-con-timeout = 10

   [junction:secureserver]
   max-cached-persistent-connections = 0
   ```

   In the example, the **worker-threads** parameter sets the maximum concurrent requests for the reverse proxy instance to 300. Therefore, the **maximum-cached- persistent-connections** for each **[junction:{junction-name}]** stanza must be less than 300.

   The **appserver1** junction allows 300 cached connections. **appserver2** allows 250 cached connections, and **secureserver** does not cache any connections.

   All other junctions are not expected to receive more than 50 concurrent requests each. Therefore, the **max-cached-persistent-connections** for the **[junction]** stanza, which is global setting of all junctions, is set to 50.

   Note: Set a high value for the persistent connections size and timeout only under the following conditions:

   - The instance is serving high concurrency requests with a high [worker-threads] value.

     Note: You must set a high value for the persistent connections timeout only under the following conditions:

     - The junctioned application server is configured with a long maximum connection timeout.

     Note: You must set a non-zero value for the persistent connections size only under the following condition:

- The reuse of a single connection reused by multiple users is not a concern.

If the junctioned server connection timeout value is low, do either of the following tasks:

- Increase the lifetime of the junctioned server connection, or
- Set the reverse proxy persistent connection timeout value slightly lower than the junctioned server connection timeout.

Important:

If the backend connection timeout is less than the reverse proxy persistent connection timeout, then reverse proxy tries to use a connection that the backend already closed. Such action can cause requests to fail at the network level.

The result is a little extra overhead as the reverse proxy opens a new connection and tries the request again.

If the junctioned server for a certain junction needs a unique connection for each request, disable the persistent connection support for this junction.

Tuning persistent connections using the Access Manager Local Management Interface

Perform the following steps to modify the Maximum Cached Persistent Connections:

1. Logon to Local Management Interface using the administrator username and password. A typical id of an administrative user in most installations is 'admin'
2. Select Secure Web Settings → Reverse Proxy.
3. Select the Reverse proxy instance
4. select Edit
5. Select the junction tab.
6. Set the new value for Maximum Cached Persistent Connections
7. Save your changes.
8. Deploy the changes.
9. Restart the reverse proxy instance to make the changes effective.

Maximum Idle persistent connections

This parameters tunes the maximum number of idle client persistent connections. The default value is 512. A low value will result in increased socket creation time between client applications and the reverse proxy.

Tuning The Front End Load Balancer on the appliance for large number of connections

Tune Front End Load Balancer on the appliance for large number of connections.

The server maxconn should be set to the maximum number of connections for that particular server, and the global maxconn should be set to the sum of all servers maxconn.

For example: if you have two appliances that you are balancing and each have 300 maximum number of connections, than you would set the server maxconn for each of them to 300, and global maxconn would be set to 600.

Tuning global maxconn using the Access Manager Local Management Interface

Perform the following steps to set the global maxconn parameter:

1. Logon to Local Management Interface using the administrator username and password. A typical id of an administrative user in most installations is 'admin'
2. Select Manage System Settings > Network Settings
3. Select Front End Load Balancer
4. Select the Advanced Tuning tab
5. Click Add
6. In the Add New Parameter window, select maxconn parameter from the Name list.
7. Enter a value for the maxconn parameter in the Value field.
8. Save and Deploy your change

Tuning server maxconn using the Access Manager Local Management Interface

Perform the following steps to set the server maxconn parameter:

1. Logon to Local Management Interface using the administrator username and password. A typical id of an administrative user in most installations is 'admin'
2. Select Manage System Settings > Network Settings
3. Select Front End Load Balancer
4. Select the Servers tab
5. Select the server to edit from the list
6. Select Edit
7. Select the Advanced Tuning tab
8. Select Add New Parameter
9. Enter maxconn for Name, and set the value
10. Save and Deploy your changes

LDAP configuration file tuning

Road map for LDAP configuration file tuning

Common to reverse proxy and authorization server

- "SSL between Security Access Manager and LDAP"
- "Load balancing between LDAP replicas"

SSL between Security Access Manager and LDAP

The communication protocol between Security Access Manager and LDAP can be either Transmission Control Protocol (TCP) or Secure Sockets Layer (SSL).

Because traffic between Security Access Manager and LDAP is light in comparison to HTTP/HTTPS traffic and because communication is over a static SSL session, the performance difference between using TCP and SSL is approximately 10 percent.

Load balancing between LDAP replicas

Security Access Manager can balance its authentication load between multiple Tivoli Directory Server servers. In environments where the Tivoli Directory Server server is the bottleneck, each additional Tivoli Directory Server server results in a linear improvement to authentication performance.

The performance improvement for adding a Tivoli Directory Server server is apparent only if the Tivoli Directory Server server is the bottleneck.

The LDAP registry configuration file controls the definition of the Tivoli Directory Server server (or servers) for use during authentication.

The ldap registry configuration file(ldap.conf) can be modified using the Security Access Manager Local Management Interface.

Statistics gathering

Process the reverse proxy logs for throughput information

The wwwlog_tput_calc.sh monitoring and diagnostic aid script can be used to process the reverse proxy www logs to extract reverse proxy throughput history information.

The script accepts an argument that is either noauth or auth.

With noauth, the script counts all pages, except pages with 401 status.

With the auth value, the script counts only pages with a 401 status, which is common for basic authentication (BA).

Viewing average response time statistics using the Access Manager Local Management Interface

The Web Reverse Proxy can be configured to record transaction logs. One of the attributes that are recorded is the average request response time. This information is recorded at a per-junction level. To view a summary of the average response time that has been recorded, use the Average Response Time widget.

Procedure

1. From the dashboard, locate the Average Response Time widget. The average response time for requests is displayed on a graph.

   Note: The widget is only displayed if one or more Reverse Proxy instances have the Flow Data function enabled.
2. Under Reverse Proxy Instances, select the instance to view the average response time statistics for.
3. Under Junctions, select the junctions to display on the graph. Each junction is represented by a separate line on the graph.
4. Under Date Range, select the duration over which the response times are recorded.

External authentication interface (EAI) server

The fastest way to authenticate a user with the Security Access Manager external authentication interface (EAI) is to return the authentication data in a reverse proxy user identity structure, rather than as a Privilege Attribute Certificate (PAC). The Privilege Attribute Certificate (PAC) is slower because the creation of the PAC is performed using the authorization Java class PDPrincipal object.

The instantiation of a PDPrincipal object results in a call to a Security Access Manager authorization server which incurs greater overhead than allowing the PAC to be created by the reverse proxy. The overhead for instantiating a PDPrincipal object is only necessary if authorization methods, such as implies, are to be used by the EAI server to make authorization decisions.

Miscellaneous tuning

Limitation on Security Access Manager users and group membership

Security Access Manager users should belong to only Security Access Manager groups.

If Security Access Manager users belong to non-Security Access Manager groups, Security Access Manager must determine whether the group is a Security Access Manager group on each and every authentication. This process has a significant impact to performance. If necessary, import the required groups to Security Access Manager.

Chapter 6. Tuning Directory Server process size limits

On AIX, Linux, and Solaris operating systems, some LDAP performance tuning tasks in this document result in process sizes that exceed the operating system default limits for process size. This chapter describes how to increase the operating system limits on process size for the environments configured with Directory Server as the remote user registry for the appliance, so that the affected processes do not run out of memory.

Memory allocation failure errors

When a process runs out of memory, the process often ends. In some cases, it leaves a core dump file, an error message, or an error log entry.

On AIX systems, the system error log might indicate that the process ended due to memory allocation failure.

Use the following command to display the error log:

```
errpt –a | more
```

Operating system process size limits

On AIX, Linux, and Solaris operating systems, each user can either inherit resource limits from the **root** user or have specific limits defined. The most useful setting to use for the process size limits is unlimited. That way, the system process size limits are defined to allow the maximum process growth.

On AIX systems, the process size limits are defined in the /etc/security/limits file. A value of –1 indicates that there is either no limit or that it is unlimited. The names of the limits to increase are **data** and **rss**. For changes to the /etc/security/limits file to take effect, the user must log out of the current login session and log back in. On AIX, some limits can apply to the **root** user.

On Solaris systems, the process size limits are defined by the **ulimit** command. You can specify a value of unlimited on the command. The names of the limits to increase are **data** and **vmemory**. By default on Solaris systems, the **root** user has unlimited access to these resources (for example, unlimited).

When setting resource limits for a process, it is important to know that the limits that apply are those that are in effect for the parent process and not the limits for the user under which the process runs.

When AIX settings take effect

In AIX, system settings take effect when you restart the UNIX shell. Operations such as **su -**, **login**, and a **process restart** picks up the new system settings, but they do not pick them up during sudo. You can also temporarily test resource limits by running the **ulimit** command and restarting the affected processes.

Chapter 7. Troubleshooting

Troubleshooting overview

When a problem occurs that appears to be related to the Tivoli Directory Server server, you must first check the following files for error messages:

- ibmslapd.log
- db2cli.log

The default location is /var/ldap for both Solaris operating environments and AIX operating systems.

You can change the location of both of these files by modifying the ibm-slapdErrorLog and ibm-slapdCLIErrors entries in the ibmslapd.conf configuration file.

Another source of diagnostic information is the db2diag file. This file is located in the DB2 instance owners home directory in a subdirectory named sqllib/db2dump/.

For example, on a UNIX system the file might be located at:
/home/ldapdb2/sqllib/db2dump/db2diag

Problem: errors when starting the Tivoli Directory Server server

Get message SQL1478W The database has been started but only one buffer pool has been activated. SQLSTATE=01626 on db2 connect to ldapdb2 or a similar message when starting the Tivoli Directory Server server:

Cause 1

On Solaris systems, this is typically due to the shmsys:shminfo_shmmax entry in /etc/system not being set high enough to support the DB2 cache.

Solution: Change the value of the shmsys:shminfo_shmmax entry in the /etc/system file and restart the system. See “Increasing the shared memory maximum (shmmax)” for more information.

Cause 2

This problem can also occur when the **UTIL_HEAP_SZ DB2** configuration parameter is set too high.

Solution: Switch to the DB2 instance owner, typically the **ldapdb2** user (su – ldapdb2) and reduce the **UTIL_HEAP_SZ DB2** configuration parameter. For example, enter:

```
db2 update db config for ldapdb2 using UTIL_HEAP_SZ 5000
```

Cause 3

This problem can occur when the DB2 buffer pool sizes are set too high.

Solution: Switch to the DB2 instance owner, typically the**ldapdb2** user, (su – ldapdb2) and reduce the buffer pool configuration parameter. For more information, see "5. Tuning the DB2 database parameters"

Problem: the wrong number of processors

The /export/home/ldapdb2/sqllib/db2dump/db2diag.log file reports a message like this one:

```
SQL8017W The number of processors on this machine exceeds the defined entitlement of
"1" for the product "DB2 Enterprise Edition". The number of processors on this machine
is "4".
```

Solution: Purchase an updates license and then use **db2licm** to update the system license. For example, enter

```
db2licm –l
```

to get the password, then enter

```
db2licm –n DB2UDBEE 4
```

to do the update. In this example, the password is DB2UDBEE and the number of processors is 4.

Problem: LDAP or DB2 fails to start

LDAP or DB2 fails to start and there is no message in the LDAP error log. It returns to the command prompt when attempting to start the Tivoli Directory Server server.

Cause: The DB2 **BUFFPAGE** parameter is set too high for the available physical memory and paging space.

Solution: Decrease the buffer pool size. See "Tuning the DB2 database parameters".

Problem: LDAP or DB2 fails to start after a DB2 restore

After a DB2 restore, LDAP, and DB2 fail to start with a message indicating that the database needs to be migrated.

Cause: See "Tuning the DB2 database parameters" for reasons why this occurs.

Solution: The workaround is to create a script that continuously loops and sets the DB2 **BUFFPAGE** parameter every 5 seconds. Run the script during the restore process. Ensure that the buffer pool is defined with a size of **–1** before running the script. For example:

```
db2 connect to ldapdb2
while [ 1 = 1 ];do
sleep 5
 db2 update database configuration for ldapdb2 using  BUFFPAGE 16000
done
```

Problem: insufficient size for the maximum shared memory

The SQL1220N The database manager shared memory set cannot be allocated. message is displayed.

Cause: An insufficient size for the shared memory maximum has been set.

Solution: See "Increasing the shared memory maximum (shmmax)

Problem: the user registry is not available

The **ldapsearch** commands return with Security Access Manager operation errors indicating that the registry is not available. The **ldapsearch** command returns with no results when results are expected.

Cause: DB2 has been forcefully stopped, using **db2 terminate**, and possibly **db2stop** and **db2start**.

Solution: Restart LDAP.

Problem: no results returned when results are expected

The **ldapsearch** commands do not fail, but return no results when results are expected.

Cause: Authentication parameters on **ldapsearch** were either not specified or incorrectly specified. For example, the **–D** and **–w** parameters were not specified.

Solution: Reissue the command with the authentication parameters specified, for example, **–D** and **–w**.

Problem: the DB2 runstat command fails

The DB2 **runstat** command fails and issues the following message:

SQL2310N The utility could not generate statistics.
Error "-1024" was returned

Cause: A DB2 connection does not exist.

Solution: Enter:

```
db2 connect to ldapdb2
```

and try again.

Problem: the server stops unexpectedly

A Security Access Manager server ends unexpectedly, but leaves no message or error log entry.

Cause: The process ran out of memory, due to lack of paging space or system process limits.

Solution: Either increase the system physical memory, the operating system paging space, or the system process limits and try again. See Chapter 6, “Tuning Directory Server process size limits” for information about increasing the system process limits.

Problem: the DB2 backup fails

The DB2 backup fails and issues this message:

```
SQL2009C There is not enough memory available to run the utility.
```

Cause: The DB2 **UTIL_HEAP_SZ** parameter is not set high enough for the backup utility.

Solution: Increase the DB2 **UTIL_HEAP_SZ** configuration parameters using the following command:

```
db2 update database configuration for ldapdb2 using UTIL_HEAP_SZ
```

Problem: the database transaction log is full

There is an LDAP or DB2 error message indicating that the transaction log for the database is full.

Cause: The **LOGFILSIZ** DB2 parameter is set too low.

Solution: Create the **LOGFILSIZ** DB2 parameter using the following command:

```
db2 update database configuration for ldapdb2 using LOGFILSIZ 10000
```

Ensure that there is enough file space in the DB2 instance owner home directory. the DB2 instance owner is typically the **ldapdb2** user

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.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing

Legal and Intellectual Property Law IBM Japan, Ltd.

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokyo 103-8510, Japan

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 might 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.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation

2Z4A/101

11400 Burnet Road

Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

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. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

© (your company name) (year). Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information in softcopy form, the photographs and color illustrations might not be displayed.

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, Acrobat, PostScript and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

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

Cell Broadband Engine and Cell/B.E. are trademarks of Sony Computer Entertainment, Inc., in the United States, other countries, or both and is used under license therefrom. Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

Printed in USA