

System Administration Certification exam 918 for IBM Informix Dynamic Server 11 prep, Part 1: IDS installation and configuration

Skill Level: Intermediate

[Manjula Panthagani \(manjulap@us.ibm.com\)](mailto:manjulap@us.ibm.com)
Advanced Support Engineer
IBM

[Rashmi Chawak \(chawak@us.ibm.com\)](mailto:chawak@us.ibm.com)
Advanced Support Engineer
IBM

[Siva K. Ane \(sane@us.ibm.com\)](mailto:sane@us.ibm.com)
Advanced Support Engineer
IBM

10 May 2007

Install and configure IBM® Informix® Dynamic Server, Version 11 (IDS 11), manage space and logs, and configure the different security options available in IDS. The first in a [series](#) of eight tutorials, use this tutorial to help prepare for the IDS 11 exam 918.

Section 1. Before you start

This tutorial explains the installation and configuration of an IBM Informix Dynamic Server 11.

About this series

Thinking about seeking certification on System Administration for IBM IDS V11 (Exam 918)? If so, you're in the right spot. This [series](#) of eight IDS certification

preparation tutorials covers all the basics -- the topics you'll need to understand before you read the first exam question. Even if you're not planning to seek certification right away, this set of tutorials is a great place to start learning what's new in IDS 11.

About this tutorial

This tutorial explains how to install and configure IBM Informix Dynamic Server 11.

Objectives

On completion of this tutorial, you should be able to:

- Understand how to install IDS 11
- Understand various configuration parameters, environments variables, network protocols, and the sqlhosts file
- See how some additional `onconfig` parameters in IDS 11 compare to previous versions
- Install the server in different modes, such as silent, GUI, and so on, and also bring down the server
- Understand each `ONCONFIG` parameter and be able to edit related parameters in the `onconfig` file based on the UNIX® operating system platform

Prerequisites

This tutorial is geared toward database administrators (DBAs) familiar with basic database concepts.

System requirements

You do not need a copy of IDS to complete this tutorial. However, you will get more out of the tutorial if you [download the free trial version](#) of IBM IDS 11 to work along with this tutorial.

Section 2. Installing IDS

You can choose from several installation methods:

- Console mode (This is the default mode -- it does not require you to add specific mode parameters in the command line for launching the installation application.)
- GUI mode
- Silent mode
- Extracting the installation with command-line script
- Invoking the JAR file directly

Installing Dynamic Server and other IBM Informix products with the default installation application

Use the installation command specific to the products you want to install:

- The `ids_install` command installs all components of IDS
- The `installserver` command installs the database server only
- The `installconn` command installs IConnect only
- The `installclientsdk` command installs the Client SDK only

Table 1. Installation options

Options	Description
<code>-gui</code>	Start the installation program in GUI mode
<code>-console</code>	Console (default) mode installation
<code>-silent</code>	Silent mode installation
<code>-record</code>	Generate a response file, a file that records your GUI or console installation settings, in order to have a customized <code>.ini</code> file for silent installation on another instance
<code>-acceptlicense = yes</code>	Accept license agreement in the silent mode
<code>-javahome dir</code>	Use specified JRE; to force the

	installation program to use the bundled JRE, use the <code>-javahome none</code> command
<code>-tempdir</code>	Use specified temporary directory
<code>-log file name</code>	Log installation program progress
<code>-is:freediskblocks</code>	Determine if there is adequate space for the product installation files prior to starting the installation
<code>-is:nospacecheck</code>	Prevent the installation program from checking if there is adequate space for product installation files. Use with caution -- if there is not enough space to extract the temporary files, the installation program will fail.

The new deployment feature allows you to record the response of your installation in a simple ASCII file that you can then use for all future silent installs. You can capture settings made in a GUI- or console-mode installation of Dynamic Server and bundled Informix products in a response file so that the same configuration can be applied later to silent installations on other instances. This process can save time if you want to replicate installation setup in other directories. Open a command-line prompt first and then the GUI or console installation application to complete this task. Keep the command-line session open until successful completion of the installation wizard that you choose.

Generate a response file and use it for silent installation

1. Pass the argument `./installserver [-gui] -record responsefile.ini` if you are installing Dynamic Server only, or `./ids_install [-gui] -record responsefile.ini` if you are installing Dynamic Server bundled with other Informix products. Do not name the response file `server.ini` or `bundle.ini`.
2. Complete installation in GUI or console mode.
3. Verify that your system generates a message verifying that creation of your response file was successful. As the root user, run the silent install, as in the following example, where `responsefile.ini` stands for the file name or file name and path:

```
./ids_install -silent -acceptlicense=yes -options responsefile.ini
```

Section 3. Configuring IDS

Steps required to configure the database server:

- Prepare space for data storage
- Set the appropriate environment variables
- Set up the sqlhosts file
- Configure the system using the configuration file in the \$INFORMIXDIR/etc directory

Preparing space for data storage

The IDS server uses two types of I/O methods:

Kernel AIO: Kernel asynchronous I/O is a method of performing non-blocking disk reads and writes through the operating system. It replaces the traditional read and write operations that require the process to wait while the data is written to disk or read from the disk. Instead, the server submits the read and write request, and can subsequently continue processing. When the I/O completes, the server is notified. Kernel AIO is available only on certain operating systems and hardware platforms. Kernel AIO is also invoked if the chunks are on a raw device (defined below). Kernel AIO threads run on CPU VP.

To find out whether the server supports Kernel AIO, check the release notes in the \$INFORMIXDIR/release directory.

AIO through AIO VPs: The server can also perform I/O through AIO VPs. The AIO VPs are responsible for performing read and write operations if Kernel AIO is not invoked. Also the AIO VPs perform I/O on all cooked files.

A *raw device* is a character-special device that you create using a UNIX utility that associates a device pathname with a device driver. This driver is a part of the operating system that translates I/O requests into instructions for the disk hardware. It is independent of the UNIX file system.

A *cooked file* is a regular file that is managed by the operating system. While the database server controls the contents of the file, it must make I/O requests to the operating system.

Preparing a raw device:

Create a new (or identify a free) partition on a disk, and issue the following commands:

```
chmod 660 <device_name>
chgrp informix <device_name>
chown informix <device_name>
```

It is not recommended to use UNIX files for your chunks, especially if your system can take advantage of Kernel AIO. However, cooked files are very easy to set up without having to investigate the availability of disk devices. If you want to use a UNIX (cooked) file for your chunk(s), you must set up a file to be used for the chunk.

Creating a cooked file:

Issue the following commands:

```
touch <filename>
chmod 660 <filename>
chgrp informix <filename>
chown informix <filename>
```

Setting up the environment

Before initializing your server, be sure that your environment includes the variables listed. Use the `env` command in UNIX to check your current environment variable setting.

Set the following environment variables before initializing a server:

Table 2. Environment variables

Variable	Description
INFORMIXDIR	Set to the directory where the IBM Informix products are installed (for example, /usr/informix)
PATH	Must include \$INFORMIXDIR/bin
INFORMIXSERVER	Set to the value of either the DBSERVERDBNAME or DBSERVERALIASES configuration parameter

Listing 1 provides an example of a setup script that sets these environment variables:

Listing 1. Sample file containing environment variables (C SHELL)

```
source ~/.env.11.10

setenv INFORMIXDIR /usr3/11.10/sqlldist
setenv TERMCAP     ${INFORMIXDIR}/etc/termcap
setenv SHELL      /bin/sh
setenv TERM       vt100
setenv INFORMIXSQLHOSTS /$INFORMIXDIR/etc/sqlhosts
setenv PATH       .:$INFORMIXDIR/bin
setenv INFORMIXSERVER menlo
setenv ONCONFIG   onconfig.11.10
```

Section 4. Managing disk space

Physical units

The database server uses the following physical units to manage disk space:

- Chunk
- Page
- Extent
- Blobpage
- Sbpage

Chunk

Chunk is a unit of disk or physical space that is assigned to the server. A chunk can be raw device (character-special device) or a UNIX cooked file. When assigning chunk to the server, you need to specify following three values:

1. **Pathname:** The path of the file or the raw device name to be used for the chunk.
2. **Offset:** The physical distance, specified in KB from the beginning of the device to where reading and writing of the device begins. If you create chunks that used cooked files, use an offset of zero; a non-zero offset is only used for raw devices.
3. **Size:** The amount of space specified in KB, allocated to be used in a raw

device from the offset or the size of the UNIX file used for the chunk.

Server has a logical limit of 32,767 chunks. However, the UNIX kernel might have a limit to the number of files a process can have open that supersedes the server limit.

Page

The basic unit of I/O that the server uses is page. Beginning with Version 10 of IDS, the page size is no longer fixed based on the product platform. Any chunk/dbspaces created after the root dbspace can have page size from 2K to 16K, using multiples of default page size. A bufferpool is created if one does not already exist to hold the pages of the size specified.

Extent

Collection of physically contiguous pages on a disk. Spaces for tables are allocated in units of extents. Extent sizes for table are specified at the time of table creation.

Blobpage

Blobpage is the basic unit of storage for blob data types stored in blobspace. The size of the blobpage can be configured to be multiple of the system page size.

Sbpage

An sbpage is the type of page that the database server uses to store smart large objects within an sbpace . Unlike blobpages, sbpages are not configurable. Sbpage is the same size as the database server page.

Logical units

The database server stores data in the following logical units:

- Dbspace
- Blobspace
- Sbspace
- Tblspace

Dbspace

Dbspace is a logical collection of one or more chunks used to store databases and tables. Each dbspace must have at least one chunk assigned to it.

Blobspace

A blobspace is a logical storage unit composed of one or more chunks that store only TEXT and BYTE data. The database server writes data stored in a blobspace directly to disk. This data does not pass through resident shared memory.

Sbospace

An sbospace is a logical storage unit composed of one or more chunks that store smart large objects. Smart large objects include BLOB, CLOB, and user-defined data types (UDTs).

Tblspace

Tblspace is a collection of all the extents allocated to store information for a particular table or index in a single dbspace. Space represented by a tblspace is not necessarily contiguous, but the space represented by any one of the extents is guaranteed to be contiguous.

Listing 2. Creating dbspace with onspaces

```
onspaces
-c
-d <dbspace>
-k <pagesize>
-m <mpathname moffset>
-o <offset>
-p <pathname>
-s <size>
-t <tempspace>
where
    spacename is the name of the dbspace to be created.
    pagesize is non-default page size for the new dbspace.
    mpathname is mirror pathname
    moffset is mirror offset
    offset is offset into the device in KB
    pathame is Path to the initial chunk
    size is Size of initial chunk in KB
    -t indicates if the dbspace created is temporary.
Example : To create a 1-million KB mirrored dbspace dbspace1
with an offset
of 200,000KB for initial(Primary) chunk and an offset of
450,000 for the
mirrored chunk.

onspaces -c -d dbpace1 -p /dev/rdsk/device1 -o 200000 -s
1000000 -m
/dev/rdsk/device2 450000
```

Listing 3. Creating blobspace

```
onspaces
-c
-b <spacename>
-g <blobpagesize>
-m <mpathname moffset>
-o <offset>
-p <pathname>
```

```
-s <size>
```

where spacename is the name of the blobospace to be created.
blobpagesize is Blobpage size in number of disk pages.
mpathname is mirror pathname.
moffset is mirror offset.
offset is Offset into the device in KB
pathname is Path to the initial chunk
size is Size of initial chunk in KB

Example : To create a 1-million KB mirrored blobospace blobosp1
with an offset of 200,000KB for both initial(Primary) and
mirrored chunk and a blobpage size of 100KB (2K system page
size) .

```
onspaces -c -b blobosp1 -g 50 -p /dev/rdsk/device8 -o 200000 -s  
1000000 -m  
/dev/rdsk/device9 200000
```

Listing 4. Creating sbospace

```
onspaces  
-c  
-S <spacename>  
-m <mpathname moffset>  
-o <offset>  
-p <pathname>  
-s <size>  
-t  
-Ms <metasize<  
-Mo <metaoffset< :  
-Df <options<
```

where spacename is the name of the blobospace to be created.
blobpagesize is Blobpage size in number of disk pages.
mpathname is mirror pathname.
moffset is mirror offset.
offset is Offset into the device in KB
pathname is Path to the initial chunk
size is Size of initial chunk in KB
-t indicates if the dbospace created is temporary.
metasize is Size of the sbospace metadata area in KB.
metaoffset is Offset of the metadata area into the
sbospace in KB
options Lists default specifications for smart large
objects stored
in the sbospace.

Example : To create a 1-million KB mirrored sbospace sbospacel
with an
offset of 200,000KB for both initial(primary) and mirror
chunks, a metadata
size of 7500 KB with 10000 KB offset and expected average smart
blobsize of 32KB.

```
onspaces -c -S sbospacel -p /dev/rdsk/device5 -o 200000 -s  
1000000 -m  
/dev/rdsk/device6 200000 -Ms 7500  
-Mo 10000 -Df "AVG_LO_SIZE=32"
```

Dropping spaces

The `onspaces` utility can be used to drop dbospace, blobospace, or sbospace from the

system command line. Before a dbspace can be dropped, all the databases and tables created in that dbspace must be dropped.

Before a blobspace can be dropped, all the tables that have a TEXT/BYTE column that reference the blobspace must be dropped.

Listing 5. Dropping spaces with onspaces

```
Example : To drop the dbspace dbpace1.
onspaces -d dbpace1
```

Adding a chunk to a dbspace or blobspace

The `onspaces` utility can be used to add chunk to a dbspace or a blobspace. Listing 6 shows the arguments used to add a chunk to a dbspace with the `onspaces` utility:

Listing 6. Adding a chunk to a Dbspace or blobspace

```
onspaces
-a <spacename>
-m <mpathname moffset>
-o <offset>
-p <pathname>
-s <size>

where
  spacename is the name of dbspace or a blobspace to which
  chunk needs
  to be added.
  mpathname is mirror pathname
  moffset is mirror offset
  offset is offset into the device in KB
  Pathname is path to the device or filename of the chunk
  Size is size of the chunk in KB

Example : To add a 500,000 KB mirrored chunk to dbpace2 with a
100,000 KB offset.

onspaces -a dbpace2 -p /dev/rdisk/device4 -o 100000 -s 500000
```

Adding a chunk to a sbpace

The `onspaces` utility can be used to add chunk to a sbpace. Listing 7 shows the arguments used to add a chunk to a sbpace with the `onspaces` utility:

Listing 7. Adding a chunk to a sbpace

```
onspaces
-a <spacename>
-m <mpathname moffset>
-o <offset>
-p <pathname>
```

```
-s <size>
-Ms <metasize>
-Mo <metaoffset>
-U
```

where

spacename is the name of sbspace to which chunk needs to be added.

mpathname is mirror pathname

moffset is mirror offset

offset is offset into the device in KB

Path to the device or filename of the chunk

Size of the chunk in KB

metasize is Size of the sbspace metadata area to be allocated in the

new chunk in KB .

metaoffset is the Offset of the metadata area into the new chunk in KB

-U Indicates that the new chunk is to contain only user data

Example : To add a 100,000KB mirrored chunk to the sbspace named sbSPACE1 with an offset of 20,000 KB for both initial and mirrored chunks and metadata size of 750KB and 1000 KB offset .

```
onspaces -a sbSPACE1 -p /dev/rdisk/chunk6 -o 20000 -s 100000 -m
/dev/rdisk/chunk7 -Ms 750 -Mo 1000
```

SQL admin API

You can use the new built-in SQL administration API functions to perform administrative tasks remotely through EXECUTE FUNCTION statements of SQL. They are the ADMIN and TASK functions. These provide SQL interface to administrative command-line utilities of IDS. These can only be called by user informix and are defined in the sysadmin database.

ADMIN or TASK functions perform a specified task which is equivalent to an administrative utility, insert a new row in the command_history table in the sysadmin database.

Listing 8. Creating a dbspace using task() function

```
EXECUTE FUNCTION task('create dbspace','dbs1',
'/dev/rdisk/device1 ', '200000',
'1000000');
Output : (expression) Space 'dbs1' added.
```

Listing 9. Dropping the dbspace using task() function

```
EXECUTE FUNCTION task ('drop dbspace', 'dbs1');
Output : (expression) Space 'dbs1' dropped.
** WARNING ** A level 0 archive will need to be done before
```

```
any chunks from
DBspace dbs1 can be reused (see Dynamic Server Administrator's
manual).
```

Listing 10. Adding mirroring to the above dbspace using task() function

```
EXECUTE FUNCTION task ( 'add mirror', 'dbs1',
'/dev/rdisk/device1', '200000',
'/dev/rdisk/device2', '450000');
Output: (expression) Mirror chunk '/dev/rdisk/device2' added to
space 'dbs1'
```

Listing 11. Creating a blobspace using task() function

```
EXECUTE FUNCTION task('create blobspace ', 'blobsp1',
'/dev/rdisk/device8 ', '50',
'200000', '1000000');
Output: (expression) Space 'blobsp1' added.
```

Listing 12. Creating a sbpace using task() function

```
EXECUTE FUNCTION task('create blobspace ', 'blobsp1',
'/dev/rdisk/device8 ', '50',
'200000', '1000000');
Output: (expression) Space 'blobsp1' added.
```

Section 5. Configuring connectivity

Connection types and communication protocols

There are three methods available for configuring client-server connectivity to an online system:

1. Through a shared memory connection. When the client application and the database server are on the same host computer, this is the preferred method of communication. The client application and the server attach to the same segment of the shared memory.
2. Through TCP/IP, using sockets or TLI programming interface. TCP/IP can

be used for both local and remote communication.

3. Through a stream pipe connection. This is a local, inter-process communication method that uses UNIX streams.

sqlhosts file

When an application attempts a connection to a database server, there is some basic information needed to make the connection. This information is written in `$INFORMIXDIR/etc/sqlhosts`, a file that must be in the `$INFORMIXDIR/etc` directory. To change location of the `sqlhosts` file, use the `INFORMIXSQLHOSTS` environmental variable. Each computer that hosts a database server or a client must have an `sqlhosts` file.

Each entry (each line) in the `sqlhosts` file contains the `sqlhosts` information for one database server. Use white space (spaces, tabs, or both) to separate the fields. Do not include any spaces or tabs within a field. To put comments in the `sqlhosts` file, start a line with the comment character (`#`). You can also leave lines completely blank for readability. Additional syntax rules for each of the fields are provided in the following sections, which describe the entries in the `sqlhosts` file. Use any standard text editor to enter information in the `sqlhosts` file.

Listing 13 depicts the sample `sqlhosts` file:

Listing 13. Sample `sqlhosts` file

```
dbservername nettype hostname servicename
menlo onipcshm india menlo
Note: This is not mandatory since it is SHM connection.
lenexa ontlitcp california cupertino
asia.1 onsoctcp node6 svc8
```

`dbservername` corresponds to the `INFORMIXSERVER` environment variable and `DBSERVERNAME` or `DBSERVERALIASES` in the `ONCONFIG` file.

The `nettype` column contains crucial information about the type of database server and how the connection is to be made. The `nettype` consists of eight letters divided into three categories.

The first two letters represent the database server product. The second three letters refer to the programming interface used for the connection. The last three letters refer to the specific protocol or IPC mechanism.

Listing 14. `Nettype` column

```
d d i i i p p
```

```
on - Dynamic Server
se - Standard Engine

ipc - IPC connection
tli - TLI connection
soc - socket connection

shm - Shared memory
str - Stream pipes
tcp - TCP/IP protocol
spx - IPX/SPX protocol
```

Hostname is the name of the local host machine.

All the service names need to have unique service numbers entered in `/etc/services`.

Listing 15. Sample `/etc/services` file

```
menlo          1543/tcp
cupertino      8262/tcp
svc8           8244/tcp
```

Section 6. Configuration file

The configuration file: Linux/UNIX.

The configuration parameters for the server are stored in a file located in the `$INFORMIXDIR/etc` directory.

Specify the name of this file by setting the `ONCONFIG` environment variable. Do not specify the full path, only the file name.

If the `ONCONFIG` environment variable is not defined, the default filename `onconfig` is used.

Example:

```
export ONCONFIG=onconfig.server1
```

The configuration file contains many different parameters that allow you to configure your server for your specific needs. Some parameters are set at the time you first set up your server and, once the server has been initialized for the first time, cannot be changed. Most parameters, however, can be modified after server initialization.

The following sections of parameters in the configuration file must be configured

before a server can be initialized because the root dbspace contains the reserved pages, information about all databases on the server, and databases that track server activity:

- Root dbspace
- Messages
- Server information

Root dbspace

Every server must have a root dbspace. Initially, the root dbspace also contains the physical and logical logs. However, these can later be moved to other dbspaces.

Listing 16. Configuring the root dbspace

```
ROOTNAME      rootdbs # Root dbspace name
ROOTPATH      /dev/online_root # Path for device containing
root dbspace
ROOTOFFSET    0 # Offset of root dbspace into device
(kilobytes)
ROOTSIZE      20000 # Size of root dbspace (kilobytes)
```

These parameters can only be modified before initializing the server for the first time. They cannot be changed once the space has been allocated for the root dbspace during server initialization.

Messages

Listing 17. Configuring message paths

```
MSGPATH      /usr/informix/online.log # System message log file
path
CONSOLE      /dev/console # System console message path
```

IDS Server provides two different destinations for server messages:

- **MSGPATH:** This parameter indicates the path and name of a file where all server messages are written. This file is created when the server is initialized for the first time, if it does not already exist.
- **CONSOLE:** This is the path specifying where the server is to write console messages. A console message is one that is of importance to the

administrator of the computer on which the server resides. For example, backup and restore requests to change the tape are sent to CONSOLE. By default, this parameter is set to the console device of the computer, but it can also be set to a file.

Server information

Listing 18. Configuring server-specific information

```
SERVERNUM          Unique id corresponding to an IDS
server
DBSERVERNAME      Name of default database server name
DBSERVERALIASES   Names of additional database server
names
```

These parameters must be set so that your server can be uniquely identified on the host computer.

Log information

Listing 19. Configuring log-specific information

```
LOGBUFF           Size in kilobytes for the three logical-log
buffers in shared memory
LOGFILES          Number of logical-log files
LOGSIZE          Size of logical-log files
PHYSBUFF         Amount of shared memory reserved for the buffers
PHYSFILE         Size of the initial physical log
PHYSDBS          Name of the dbspace in which the physical log
resides
```

Logical logs

The logical log files are collections of contiguous pages on disk that are used to store transaction records for the server. These transaction records are used to track all the changes made to databases that were created with logging. All databases share the same set of logical log files. Each server must have at least three logical log files.

Adding logical log files manually

You might add logical-log files manually for the following reasons:

- To increase the disk space allocated to the logical log

- To change the size of your logical log files
- To enable an open transaction to complete
- As part of moving logical log files to a different dbSPACE

The two ways you can add a logical log file are:

1. To the end of the file list using the `onparams -a` command
2. After the current logical log file using the `onparams -a -i` command

The following command adds a logical log file to the end of the log file list in the `logSPACE` dbSPACE, using the log file size specified by the `LOGSIZE` configuration parameter:

```
onparams -a -d logSPACE
```

The following command inserts a 1000KB logical log file after the current log file in the `logSPACE` dbSPACE:

```
onparams -a -d logSPACE -s 1000 -i
```

To add a logical log file with a new size (in this case, 250KB), execute the following command:

```
onparams -a -d logSPACE -s 250
```

You can drop a logical log file using:

```
onparams -d -l lognum -y
```

You can move logical log files by:

- Dropping logical log files from their current dbSPACE
- Adding the logical log files to their new dbSPACE

Strategy for estimating the size and number of the logical logs

It is easy to manage a few large log files than many small log files. If there are not enough log files, frequent checkpoints will be triggered, which in turn impacts performance. For applications that generate a small amount of log data, start with 10 log files of 10MB each, and for applications that generate a large amount of log data, start with 10 log files with 100MB. If smart large objects in blobspaces are updated frequently, you need frequent log backups to free up the space in the blobspace.

The following expression provides an example total-log-space configuration, in kilobytes:

```
LOGSIZE = (((connections * maxrows) * rowsize) / 1024) / LOGFILES
```

Recovery Time Objective (RTO) policy determines the tolerance for loss of data in case of a server crash. If this RTO policy is required,

1. Use automatic log backups that will back up logs whenever a log file fills up
2. Use the scheduler to create tasks that automatically back up up new log data at timed intervals
3. If the log space fills up before the log files are backed up, add a new log file to allow transaction processing to continue
4. Use the scheduler to add a task to detect the above situation and add logs automatically

Physical log

The server has a special log that is used for automatic recovery purposes. This log is called the physical log. The physical log is a collection of contiguous pages on disk.

When a page is read into a shared-memory buffer and modified by a user, a copy of the page in its original condition is written to the physical log. This copy of the page is known as a before image (the copy of the page before it was changed). Only the first change to a page in a buffer causes a before image to be written to the physical log. Any subsequent changes to that same page do not cause additional before images to be written to the physical log. These before images are used by an automatic-recovery mechanism.

You can move the physical log location and size using `onparams`.

The following command moves the physical log to the dbspace `dbspace1` and

resizes it to 3000KB:

```
onparams -p -d dbSPACE1 -s 3000
```

Estimating the size of the physical log

PHYSFILE configuration parameter specifies the size of the physical log. The size of physical log depends on:

1. The rate at which transactions generate physical log activity.
2. Use of RTO_SERVER_RESTART configuration parameter to specify a target amount of time for fast recovery.

Checkpoints are triggered when physical log is 75 percent full and processing needs to complete before the rest of the physical log is used, so the database server blocks all the transactions to avoid filling up the physical log. In order to avoid transaction blocking, make sure the database server has enough physical log space to contain all the transaction activity that occurs during the checkpoint processing.

If you specify a value for the RTO_SERVER_RESTART configuration parameter, server monitors the workload and triggers checkpoints to meet the RTO policy. This generates some physical log activity. The extra logging is used to assist the bufferpool during fast recovery so that log replay performs optimally. If the physical log is considerably larger than the combined sizes of all buffer pools, page flushing and page faulting occurs during fast recovery. The page flushing and page faulting substantially reduces fast recovery performance, and the database server cannot maintain the RTO policy.

For systems with less than 4GB of buffer pool space, the physical log can be sized at 110 percent of the combined size of all the buffer pools. For larger buffer pools, start with 4GB of physical log space and then monitor checkpoint activity. If checkpoints occur too frequently and appear to impact performance, increase the physical log size.

You can use the `onstat -g ckp` command to display configuration recommendations.

New onconfig parameters for IDS 11

Table 3. Onconfig parameters

Configuration parameter	Description/comment
RTO_SERVER_RESTART	Enables you to use recovery time objective (RTO) standards to set

	the amount of time, in seconds, that Dynamic Server has to recover from a problem after you restart Dynamic Server and bring the server into online or quiescent mode
RAS_PLOG_SPEED	The rate at which the physical log can be recovered during fast recovery
RAS_LLOG_SPEED	The rate at which the logical log can be recovered during fast recovery. Not configurable. IDS updates these values to reflect what the real recovery speed is. (The units are pages per second.)
AUTO_CKPTS	The rate at which the physical log can be recovered during fast recovery
RAS_PLOG_SPEED	Enables or disables automatic checkpoints
AUTO_LRU_TUNING	Enables or disables automatic LRU tuning
AUTO_AIOVPS	Enables or disables the ability of the database server to automatically increase the number of AIO VPs and flusher threads when the server detects that AIO VPs are not keeping up with the I/O workload
SQLTRACE	Controls the default behavior such as the number of SQL statements to trace and the tracing mode of the query drill-down feature
EXPLAIN_STAT	Enables or disables the inclusion of a query statistics section in the explain.out file that the <code>SET EXPLAIN</code> statement of SQL or the <code>onmode -Y session_id</code> command can display
USELASTCOMMITTED	Specifies whether the database server uses the last committed version of the data when a lock occurs
SHMVIRT_ALLOCSEG	Specifies a threshold at which Dynamic Server should allocate server memory and the alarm level activated if the server cannot allocate the new memory

	segment
ENCRYPT_HDR	Enables or disables encryption between servers in an HDR pair
LOG_INDEX_BUILDS	Set to one to enable the logging of the index pages during the create index statement. Required on the primary node when using Remote Standalone Secondary (RSS) nodes.
ENCRYPT_SMX	0. Do not use Encryption on SMX connection 1. Negotiate encryption on SMX connection 2. Must use encryption on SMX connection

Section 7. Configuring security

Label-based access control security feature

Label-based access control (LBAC) is a means by which the database system controls access to table objects. The database system controls access by attaching security labels to the table objects and comparing this with the security labels granted to users attempting to access that object. It grants access if the two match, denies access otherwise.

There are three types of security labels:

1. **Row security label:** A security label associated with a data row or record in a database table
2. **Column security label:** A security label associated with a column in a database table
3. **User security label:** A security label granted to a database user. A user can have labels for read access or write access.

A security label can contain one or more security label components. There are three types of security label components:

1. **Set:** A set is a collection of elements where the order in which those elements appear is not important
2. **Array:** An array is an ordered set. In an array, the order in which the elements appear is important because it denotes the degree of sensitivity of the data
3. **Tree:** A tree represents a hierarchy that can have multiple nodes and branches. For example, trees can be used to represent organizational charts and to identify different departments within the organization.

A security policy defines the set of security label components that make up a security label. DBSECADM is required to manipulate LBAC objects.

Pluggable Authentication Modules for systems running on UNIX or Linux

A Pluggable Authentication Module (PAM) was originally developed by Sun Microsystems to support different authentication modules. PAM allows system administrators to implement different authentication methods for different applications.

Authentication services are attached at the application level. For example, the authentication scheme can be different for a database than the UNIX login program.

Another feature of PAM is module stacking, where many modules can be stacked one over another, providing authentication to the application in many ways. PAM provides a set of APIs to support authentication -- account management, session management, and password management.

The system administrator can enable or disable the use of PAM.

Pluggable Authentication Modules are supported in both 32-bit and 64-bit modes on Solaris, Linux, HP-UX, and AIX®.

LDAP authentication support on Windows

You can use the LDAP Authentication Support module when you want to use an LDAP server to authenticate your system users. LDAP on Windows is configured similar to Pluggable Authentication Module on UNIX and Linux, but does not support the module stacking feature. It only provides single-module authentication.

The authentication DLL is located in the %INFORMIXDIR%\dbssodir\lib\security

directory. The configuration parameters are listed in the %INFORMIXDIR%\dbssodir\pam.conf file. The source code for LDAP Authentication Module and samples of the required configuration files are included in the %INFORMIXDIR%\demo\authentication directory.

The system administrator can enable or disable the authentication.

Section 8. Summary

In this tutorial, you learned to install and configure IDS, manage space and logs, and configure the different security options available in IDS.

You learned about the different options available during installation and about the new deployment feature that allows you to record the response of your installation in a simple ASCII file, which can then be used for all future silent installs.

This tutorial talks about the steps involved in configuring a simple database server, like preparing space for data storage, making appropriate server entries into the sqlhosts file, setting the environment variables necessary for a client application to connect to the database server, and configuring the system using the configuration file. It also lists the new `ONCONFIG` parameters in IDS V11.

This tutorial discussed the logical and physical units of storage, adding and dropping dbspaces, sbspaces, and blobspaces using the `onspaces` command and sql Admin API. It also described how to manage logs using the `onparams` command as well as the strategies to determine the number and sizes of the logical and physical logs.

Additionally, this tutorial introduced you to the new feature in IDS V11, LBAC security, and other security options available.

[Part 2](#) of the tutorial series introduces you to the monitoring utilities available in IDS V11.

Resources

Learn

- [developerWorks Informix zone](#): Read articles and tutorials and connect to other resources to expand your Informix skills.
- [IBM Informix Dynamic Server Information Center](#): Learn more about Informix.
- [IBM Informix Dynamic Server 11 BETA Information Center](#): Learn more about IDS 11.
- [developerWorks IDS Experts blog](#): Find technical notes on Informix Dynamic Server by a worldwide team of Development and Technical Support engineers.
- [IBM Information Management certification page](#): Learn more about resources for IDS certification.
- [developerWorks Information Management zone](#): Learn more about Information Management. Find technical documentation, how-to articles, education, downloads, product information, and more.
- Stay current with [developerWorks technical events and webcasts](#).
- [Technology bookstore](#): Browse for books on these and other technical topics.

Get products and technologies

- [Informix Dynamic Server Enterprise Edition V10.0](#): Download a free trial version.
- [Informix Dynamic Server 11](#): Download the free trial version to work along with this tutorial.
- [IBM product evaluation versions](#): Download and get your hands on application development tools and middleware products from Information Management, Lotus®, Rational®, Tivoli®, and WebSphere®.

Discuss

- [Participate in the discussion forum for this content](#).
- Check out [developerWorks blogs](#) and get involved in the [developerWorks community](#).

About the authors

Manjula Panthagani



Manjula Panthagani is an Advanced Support Engineer for Informix Dynamic Server at IBM. She has been in this position supporting IDS for over seven years and participated in the development of the IDS Certification exam.

Rashmi Chawak



Rashmi Chawak is an Advanced Support Engineer for IDS, IBM. She has been in this position supporting IDS for over seven years and has worked on porting IDS product on various platforms.

Siva K. Ane



Siva Ane is an Advanced Support Engineer for IDS, IBM. He has been in Platform Engineering for nine years prior to this position.

Trademarks

IBM, Informix
UNIX