

Server Management

Presented by DB2 Developer Domain

<http://www7b.software.ibm.com/dmdd/>

Table of Contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

1. Introduction	2
2. DB2 instances	4
3. DB2 client/server connectivity	12
4. DB2 security	25
5. Scheduling jobs	37
6. Using notification logs	41
7. Conclusion	43

Section 1. Introduction

What this tutorial is about

This tutorial introduces skills you must have to properly manage a DB2® server. This is the first tutorial in a series of six to help you prepare for the DB2 V8.1 for Linux, UNIX®, and Windows™ Database Administration Certification (Exam 701). The material in this tutorial primarily covers the objectives in Section 1 of the exam, entitled "Server Management." You can view these objectives at:

<http://www.ibm.com/certify/tests/obj701.shtml>.

You do not need a copy of DB2 Universal Database to complete this tutorial. However, you can download a free trial version of *IBM DB2 Universal Database™* Enterprise Server Edition if you'd like.

In this tutorial, you will learn:

- Advanced topics about DB2 instances
- How to manage and configure DB2 instances
- How to configure client/server connectivity by using DB2 tools
- How DB2 authentication works
- How to control DB2 authorizations and privileges
- When and how to use the DB2 *force* command
- How to schedule jobs to run unattended
- What you can get out of the DB2 notification log

Who should take this tutorial

To take the DB2 UDB V8.1 Database Administration exam, you must have already passed the DB2 UDB V8.1 Family Fundamentals exam (Test 700). You can use the [DB2 Family Fundamentals tutorial series](#) to prepare for that test. It is a very popular tutorial series that has helped many people understand the fundamentals of the DB2 family of products.

Although not all materials discussed in the Family Fundamentals tutorial series are required to understand the concepts described in this tutorial, you should at least have a basic knowledge of:

- DB2 UDB products
- DB2 tools

- DB2 instances
- Databases
- Database objects

This tutorial is one of the tools to help you prepare for Exam 701. You should also review the resources at the end of this tutorial for more information about DB2 server management (see [Resources](#) on page 43).

About the author

Clara Liu works as a DB2 UDB consultant at the IBM Toronto Laboratory. As a member of the Data Management Channel Development Team, she works closely with IBM business partners and global system integrators. Clara specializes in database application development and integration of new technologies with DB2. She teaches DB2 UDB certification courses to IBM business partners and at conferences. She co-authored the book *DB2 SQL Procedural Language for Windows, UNIX, and Linux* (Prentice Hall, 2002). You can reach Clara at claraliu@ca.ibm.com.

Section 2. DB2 instances

Creating and dropping an instance

A DB2 *instance* is a logical context in which DB2 commands and functions are executed. You can think of an instance as a service or a daemon process that manages access to database files. More than one instance can be defined on a server machine. Each instance is independent of the others, meaning that all instances can be managed, manipulated, and tuned separately.

To create an instance in Windows, simply issue this command:

```
db2icrt instance_name
```

In UNIX, you must also provide a user ID that will be used to create fenced user-defined function and stored procedure processes, like so:

```
db2icrt -u fenced_user_ID instance_name
```

User-defined functions and stored procedures, by default, are created in fenced mode so that these processes run in a different address space as the DB2 system controller process, *db2sysc*. This protects the database manager from being accidentally or maliciously damaged by any user-defined routines.

To drop an instance, disconnect all database connections, and stop the instance, issue this command:

```
db2idrop -f instance_name
```

Listing, migrating, and updating a DB2 instance

To list DB2 instances that exist on a server, use this command:

```
db2ilist
```

Instance migration is required if a newer version of DB2 UDB is installed or if an instance is to be migrated to a 64-bit instance. On Windows, instance migration is done implicitly during the necessary migration process. On UNIX, use this command to

migrate an existing instance explicitly:

```
db2imigr instance_name
```

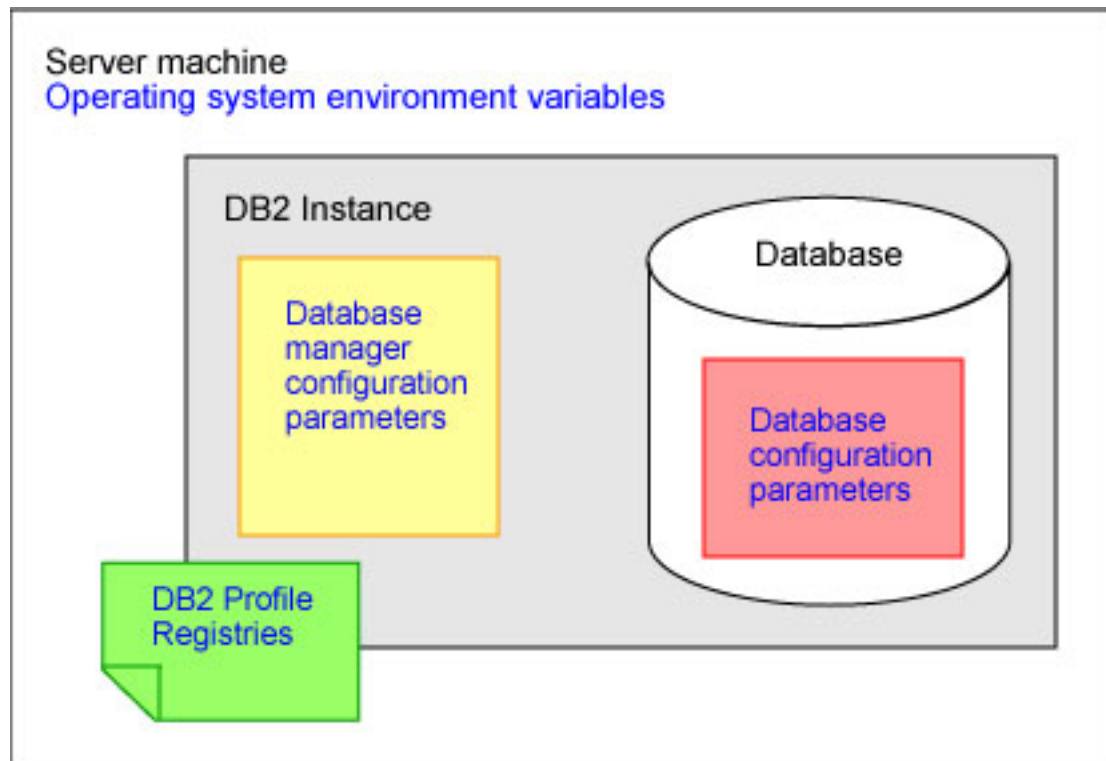
When certain product options or fix patches are installed, existing DB2 instances and their associated databases need access to new functions. Use this command to update an instance:

```
db2iupdt instance_name
```

Setting up the DB2 environment

Proper setup of the DB2 environment is very important because it controls how DB2 operates and functions. The DB2 environment is made up of:

- DB2 profile registries
- Operating system environment variables
- DB2 database manager configuration parameters
- DB2 database configuration parameters



Setting profile registries

DB2 profile registries are DB2-specific variables that affect the management, configuration, and performance of the DB2 system. You usually need to stop and restart an instance for changes made to the DB2 profile registries to take effect.

To list all supported DB2 profile registries:

```
db2set -lr
```

To set a DB2 profile registry:

```
db2set registry_variable=value
```

Note that there are no spaces between the variable name, the equals sign, and the variable value. Here's an example:

```
db2set DB2COMM=TCPIP,APPC
```

To reset a DB2 profile registry to its default value, simply use the same command as above but do not specify any value:

```
db2set registry_variable=
```

To display all the DB2 profile registries currently set on the server, issue this command:

```
db2set -all
```

You should get results that look something like this:

```
[e] DB2PATH=C:\SQLLIB  
[i] DB2INSTPROF=C:\SQLLIB  
[i] DB2COMM=TCPIP,APPC  
[g] DB2SYSTEM=DB2NTSERV  
[g] DB2PATH=C:\SQLLIB  
[g] DB2ADMINSERVER=DB2DAS00
```

Indicators surrounded by the square brackets ([]) represent the scope of the registry profile, as follows:

- [e] represents a registry setting for the current session or environment
- [u] represents a user-level registry
- [n] represents a node-level registry
- [i] represents a instance-level registry
- [g] represents a global-level registry

Setting system environment variables

Most DB2 environment settings are controlled by the DB2 profile registry. Those that are not stored in the profile registry are referred to as the *operating system environment variables*. Commands for setting the system variables will vary depending on the platforms and UNIX shells you are using.

Here are some examples:

- On Windows: `set DB2INSTANCE=PROD`
- On UNIX C shell: `setenv DB2INSTANCE PROD`
- On UNIX Korn shell: `export DB2INSTANCE=PROD`

DB2INSTANCE is an important system variable to know about. It specifies the current session's default instance. Once this variable is set, all subsequent DB2 commands are executed to the scope of the instance specified.

To find out which instance you are in:

```
db2 get instance
```

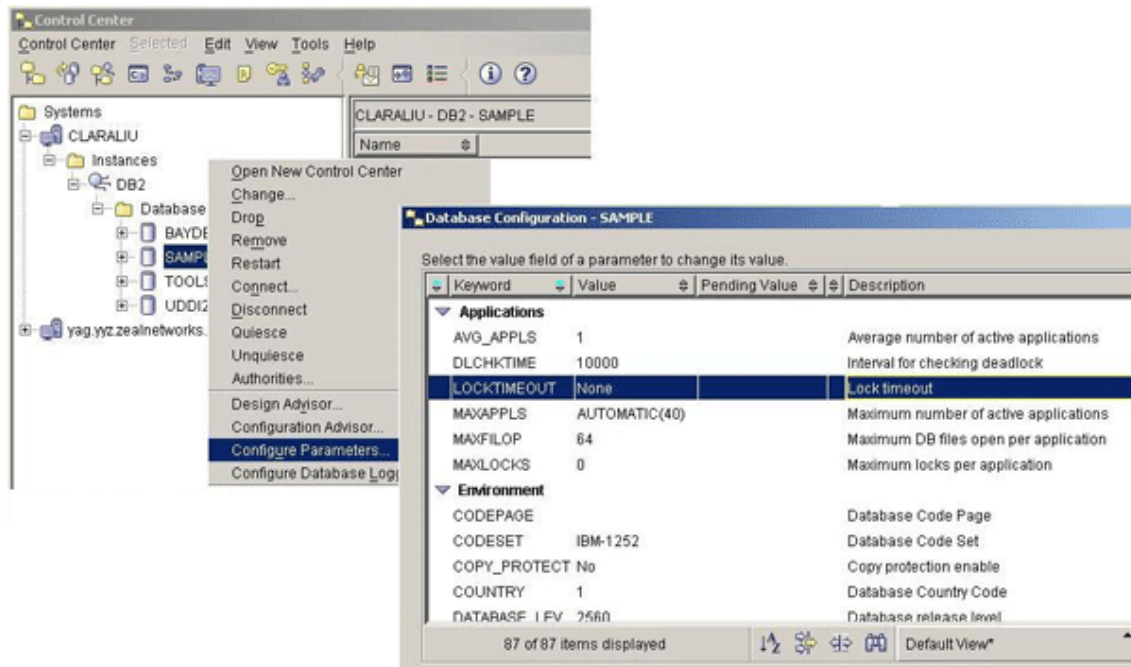
The output from this command will look something like this:

```
The current database manager instance is: DB2
```

Setting configuration parameters

There are *instance* and *database* levels of configuration parameters that can be changed to tune the behavior of DB2. Refer to the [Monitoring DB2 activity](#) tutorial for more information about the parameters. The instance (also referred to as the *database manager*) and database configuration parameters can be viewed using the DB2 Control Center and DB2 commands.

At the Control Center, right click on the instance or database for which you would like to work with and select **Configure Parameters**. You will get a list of configuration parameters with descriptions and the current and pending values.

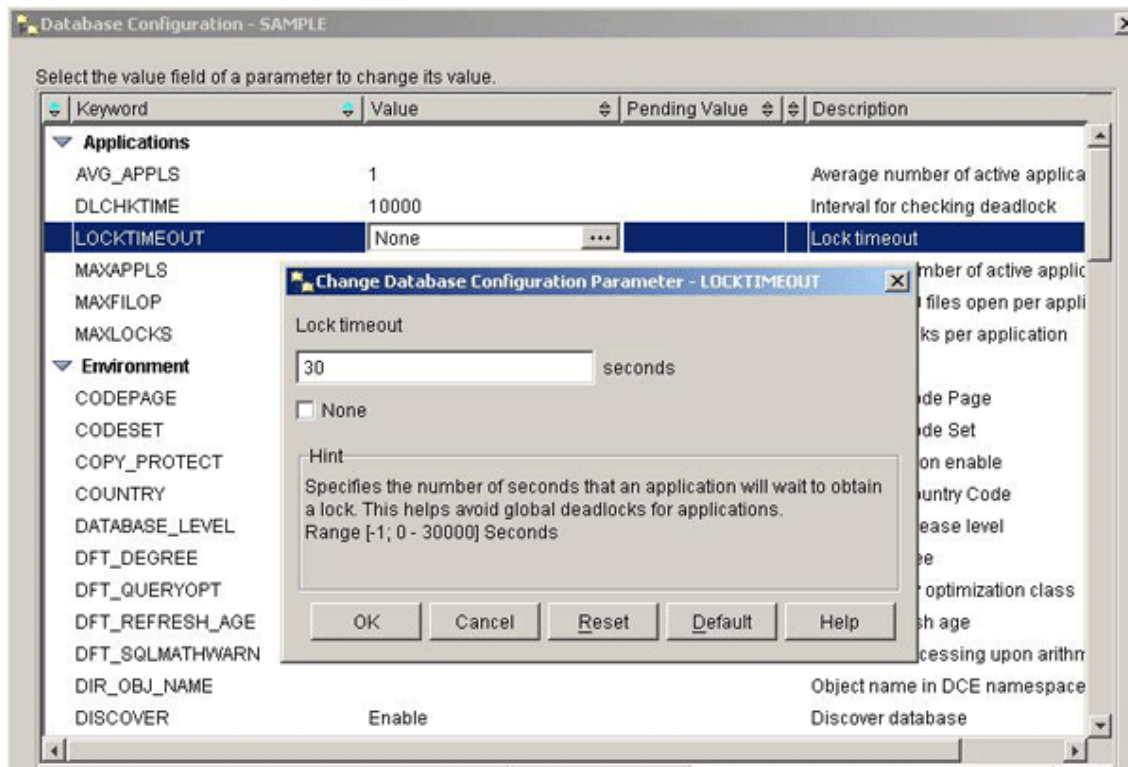


You can also get the same output by using the DB2 commands:

```
db2 get database manager configuration
```

db2 get database configuration for *database_name*

To update the values of the database manager or database configuration parameters at the Control Center, open the DBM or DB Configuration window. Find the parameter you want to change and double click on its value. The '...' icon will open a form that contains hints on how to set the variable and explanations of what ranges are allowed.



Once you click OK, you are returned to the Configuration window. The Pending Value Effective column tells you when the new value will take effect. If the field reads "After Instance Restart", then the pending value will not be set to the value you selected until the instance is restarted. Otherwise the new value takes effect immediately.

The following commands can be used to set the values of the database manager or database configuration parameters respectively:

```
db2 update database manager configuration using parameter new_value
db2 update database configuration for database_name using parameter new_value
```

If the parameter cannot take effect immediately, a warning message appears:

```
SQL1362W One or more of the parameters submitted for immediate modification
were not changed dynamically. Client changes will not be effective until the next time
the application is started or the TERMINATE command has been issued. Server changes
will not be effective until the next DB2START command.
```

Setting configuration parameters online

Over 50 configuration parameters can be set online while an instance or a database is still running. By default, changes to these online configurable parameters will take effect immediately where possible. For example, if the value of *sortheap* is changed, all new SQL requests will use the new value. To specify this immediate behavior explicitly, append the `immediate` keyword to the update command:

```
db2 update database manager configuration using parameter new_value
                                             immediate
```

```
db2 update database configuration for database_name using parameter new_value
                                             immediate
```

If you choose to defer the changes until the instance is restarted or until the database is activated, specify the `deferred` keyword instead:

```
db2 update database manager configuration using parameter new_value
                                             deferred
```

```
db2 update database configuration for database_name using parameter new_value
                                             deferred
```

You'll sometimes want to find out what changes have been made and deferred. To show the current and pending values of the database manager configuration parameters, attach to the instance first, then specify the `show detail` option in the `get database manager configuration` command, like so (note that *instance_name* is the value set by the system environment variable *DB2INSTANCE*):

```
db2 attach to instance_name
```

```
db2 get database manager configuration show detail
```

Similarly, to list the current and pending values of the database configuration parameters, connect to the database first and then use the `show detail` option:

```
db2 connect to database_name
```

```
db2 get database configuration for database_name show detail
```

Pending values are listed under the Delayed Value column, as illustrated below.

```

C:\>db2 connect to sample

Database Connection Information
Database server      = DB2/NT 8.1.0
SQL authorization ID = CLARALIU
Local database alias = SAMPLE

C:\>db2 get database configuration show detail

Database Configuration for Database

Description                                     Parameter      Current Value      Delayed Value
-----
Database configuration release level             = 8x0a00
Database release level                           = 8x0a00
Database territory                               = US
Database code page                               = 1252
Database code set                                = IBM-1252
Database country/region code                    = 1
Dynamic SQL Query management                    <DYN_QUERY_MGMT> = DISABLE          DISABLE
Discovery support for this database              <DISCOVER_DB>    = ENABLE           ENABLE
Default query optimization class                <DFT_QUERYOPT>  = 5                5
Degree of parallelism                          <DFT_DEGREE>    = 1                1
Continue upon arithmetic exceptions             <DFT_SQLMATHWARN> = NO              NO
Default refresh age                            <DFT_REFRESH_AGE> = 0                0
Number of frequent values retained              <NUM_FREQUVALUES> = 10              10
Number of quantiles retained                   <NUM_QUANTILES> = 20              20
Backup pending                                 = NO

```

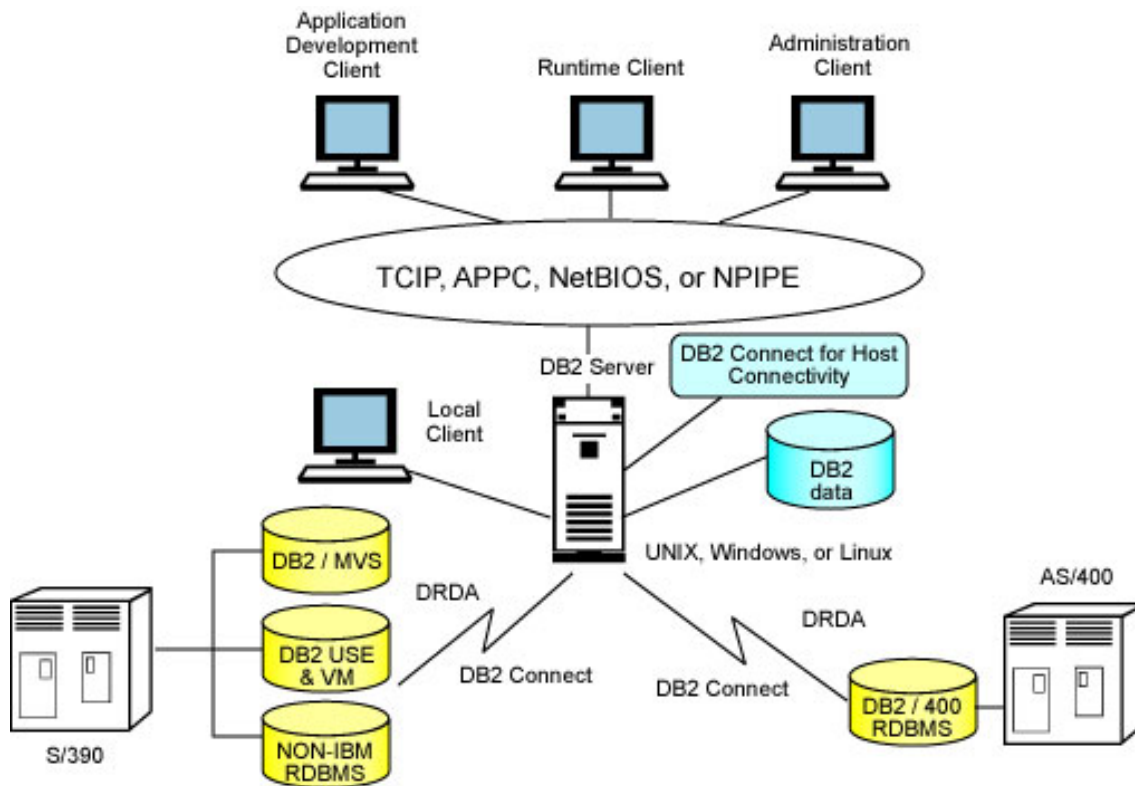
Section 3. DB2 client/server connectivity

DB2 client/server environment

Several communication protocols are supported for DB2 client/server connectivity:

- TCP/IP
- APPC
- NetBIOS
- NPIPE

The DB2 Connect software, which utilizes the Distributed Relational Database Architecture (DRDA), is required for connections to host databases such as DB2 for z/OS and OS/390 or DB2 for iSeries.



Preparing DB2 server for remote connections

Before DB2 clients can connect to a database, you must ensure that the server-side communications are set up properly. To prepare a server for TCP/IP and NetBIOS connections:

1. Set the DB2 profile registry, *DB2COMM*, to enable the instance for the specified communication protocols, like so:

```
db2set DB2COMM=TCPIP,NETBIOS
```

2. Set the necessary information for each supported protocol in the database manager configuration file.

For TCP/IP, assign a port number to each instance that is TCP/IP enabled. A file called *services* contains services defined on the system and their port numbers. The location of the file will depend on your platform. For example, on UNIX, it is usually stored in */etc*.

Since a port number can only be used by a single service, it is recommended that you use the *services* file as a central place to maintain a list of all services and their associated port numbers. To reserve TCP port 50000 for the service named *db2icdb2*, append the following line to the *services* file:

```
db2icdb2      50000/tcp
```

Update the database manager configuration file so that DB2 will use the port number associated with the service *db2icdb2* for the instance you are working on:

```
db2 update database manager configuration using svcename db2icdb2
```

If you choose not to use the *services* file, simply update *svcename* with the correct port number:

```
db2 update database manager configuration using svcename 50000
```

For NetBIOS, enter the NetBIOS workstation name (*nname*) in the database manager configuration file:

```
db2 update database manager configuration using nname DB2NTSERV
```

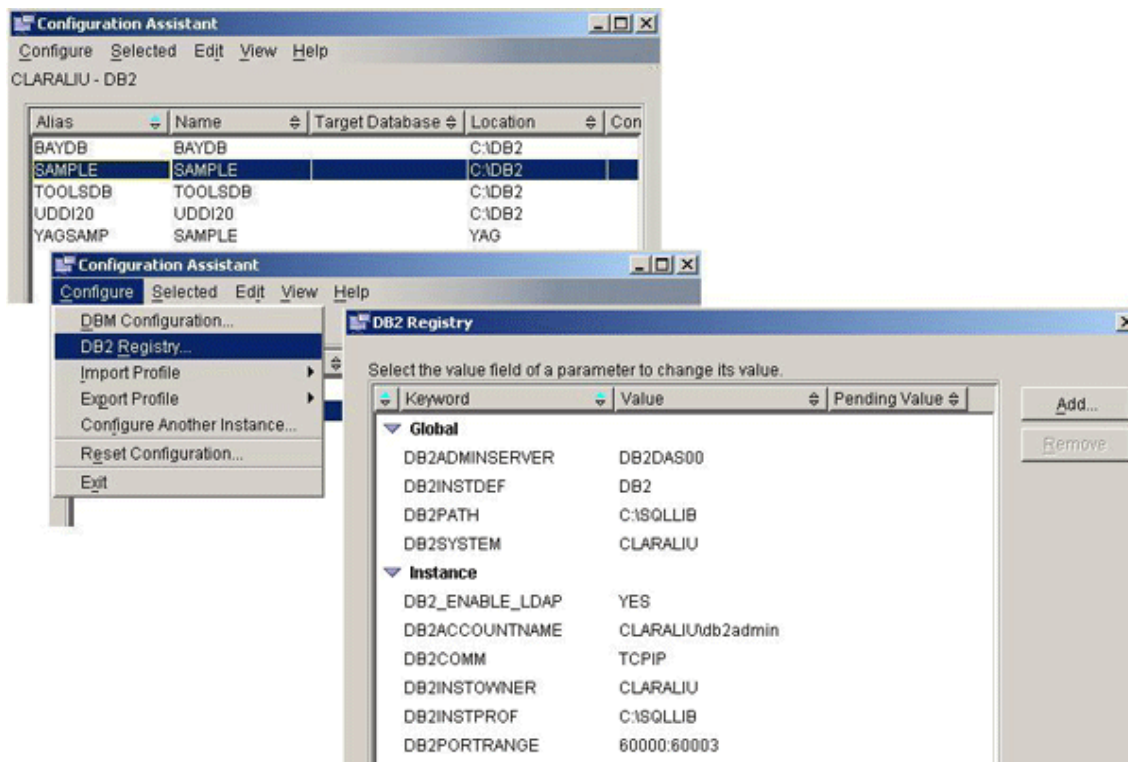
3. The parameters *svcename* and *nname* are not configurable online. Stop and start the instance so that the new values can be used:

```
db2stop  
db2start
```

Using the DB2 Configuration Assistant

The DB2 Configuration Assistant provides user-friendly wizards and a graphical interface for configuring the environment you or your applications will be using. From the Configuration Assistant, you can:

- Add new database connections
- Update database connectivity information
- View and update database manager configuration parameters
- View and update DB2 profile registries
- Bind applications to a database
- Update the Call Level Interface (CLI) settings

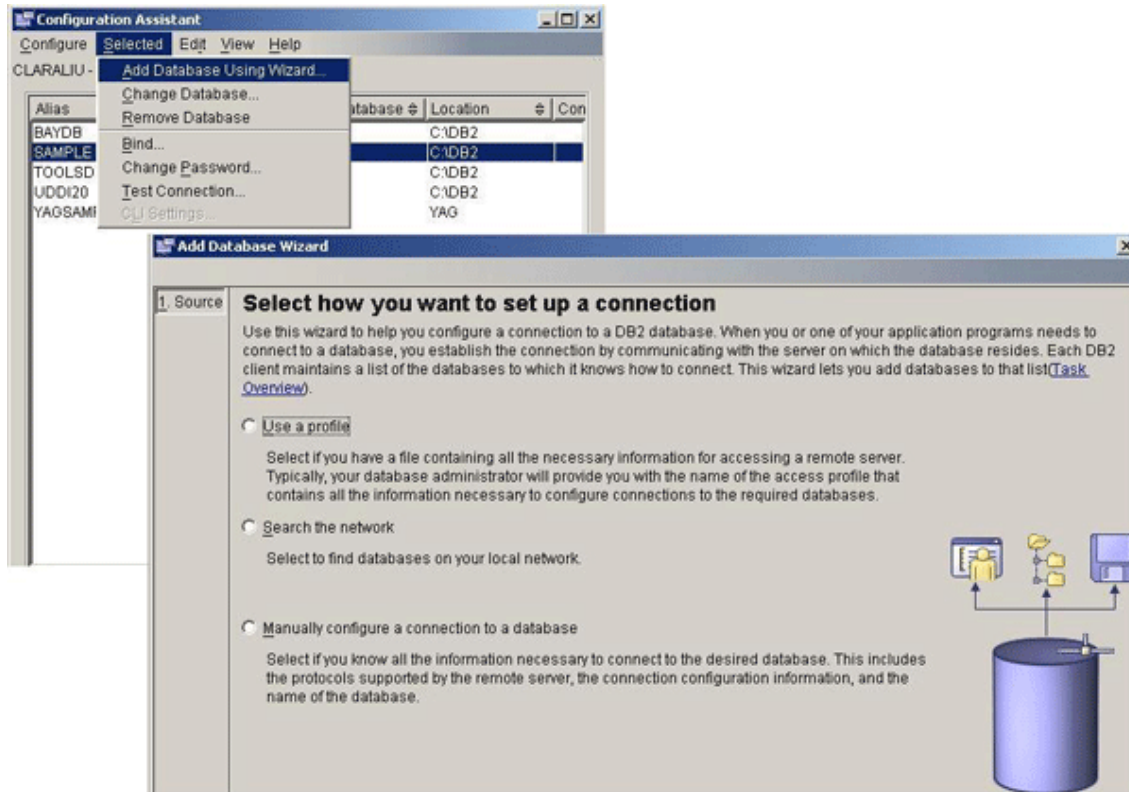


Three ways to configure database connectivity

There are three options available in the DB2 Configuration Assistant for setting up a

database connection, you can:

- Use DB2 Discovery to search the network
- Use DB2 access profiles
- Configure the connection manually

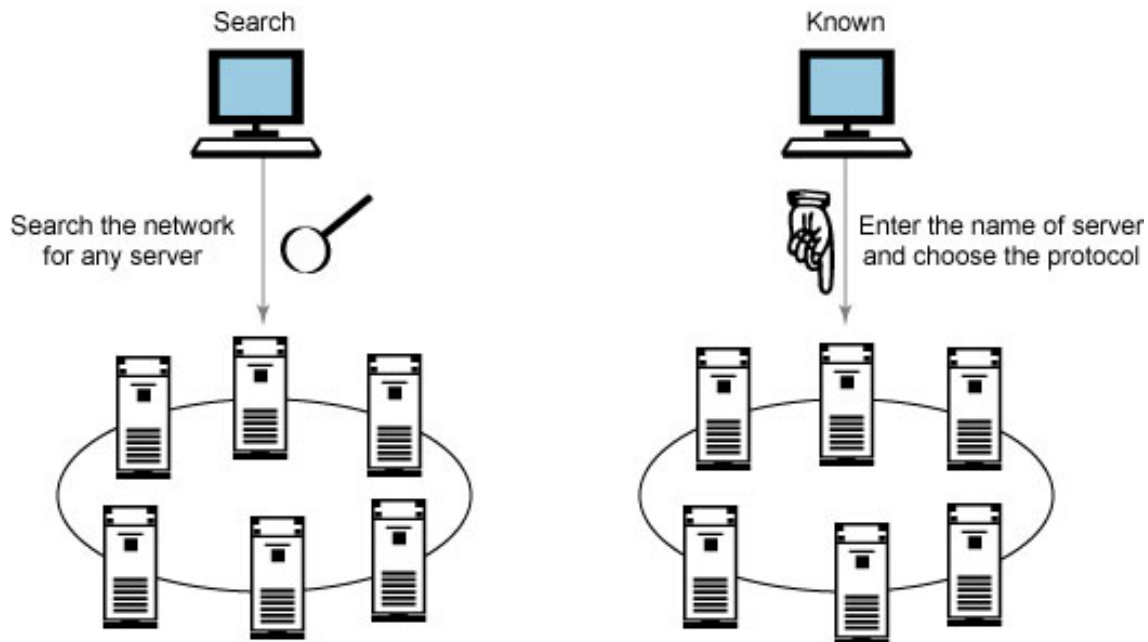


Configuring database connectivity automatically with DB2 Discovery

DB2 Discovery searches and locates DB2 servers on the network. You can choose to use either the *search* or *known* discovery methods.

The search method searches the network for any DB2 servers. DB2 Discovery performs a search up to the first router from where the search request is issued. Therefore, this method may take some time to return a result.

If you know some information about the DB2 server you want to locate, use the known method and provide information such as the database or server name to limit the search.



Sometimes you don't want certain DB2 servers, instances, or databases to be discoverable. For example, imagine a DB2 server containing a production instance and a development instance. In the development instance, two databases, ACCT and HUMRES, are defined. The DBA would like to keep the production instance from being discovered and allow only the ACCT database in the development instance to be discovered. A few configuration parameters are available to allow such behavior.

A DB2 server is searchable only if its DB2 Administration Server (DAS) service is running and the `discover` parameter is set to `search`:

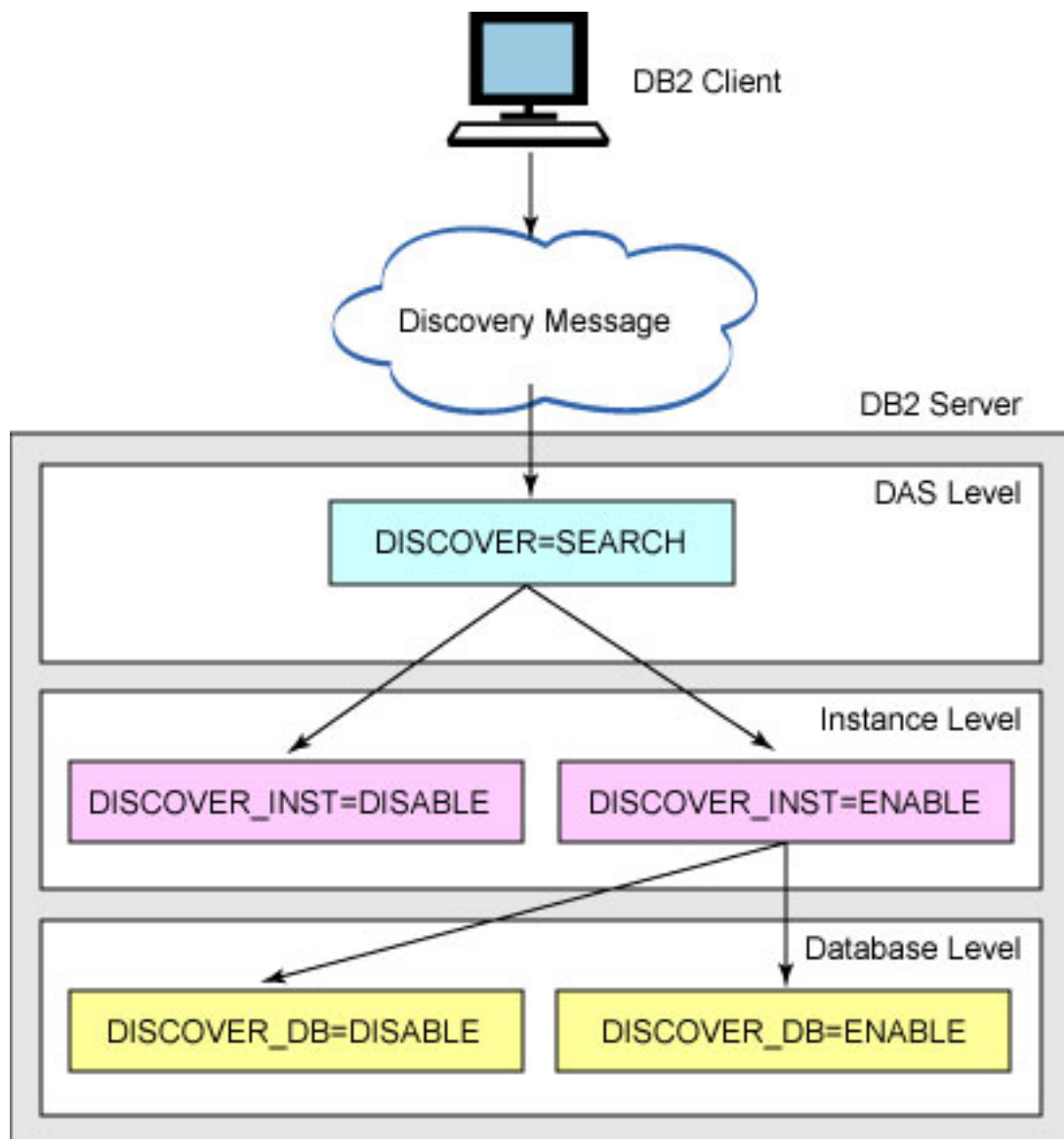
```
db2admin start
db2 update admin configuration using discover search
```

You can then control which instance is discoverable by enabling the `discover_inst` database manager configuration parameter:

```
db2 update database manager configuration using discover_inst enable
```

At the database level, there is a similar configuration parameter, `discover_db`, that can enable or disable database discovery:

```
db2 update database configuration for database_name using discover_db enable
```



It is important to point out that disabling discovery at the DAS, instance, or database level does not restrict DB2 clients from setting up database connectivity through other methods (which will be discussed in the next two panels). DB2 clients can still connect to a remote database even though its database configuration `discover_db` is disabled.

Configuring database connectivity automatically with DB2 access profiles

What would you do if you needed to set up DB2 client/server connectivity for 1,000 or

more workstations? You could go to each workstation and use the discovery method from the Configuration Assistant, but that task would probably take you a long time to complete. In such a situation, you should consider using a *DB2 access profile*.

An access profile contains information that a client needs for configuring connectivity with a DB2 server. There are two types of access profiles:

- A *server access profile* is generated on a DB2 server. It contains information about all or selective instances and databases that are defined on the server.
- A *client access profile* is generated on a DB2 client. It contains information about instances (also called *nodes*) and databases already cataloged on the client.

Let's walk through the DB2 access profile approach step by step.

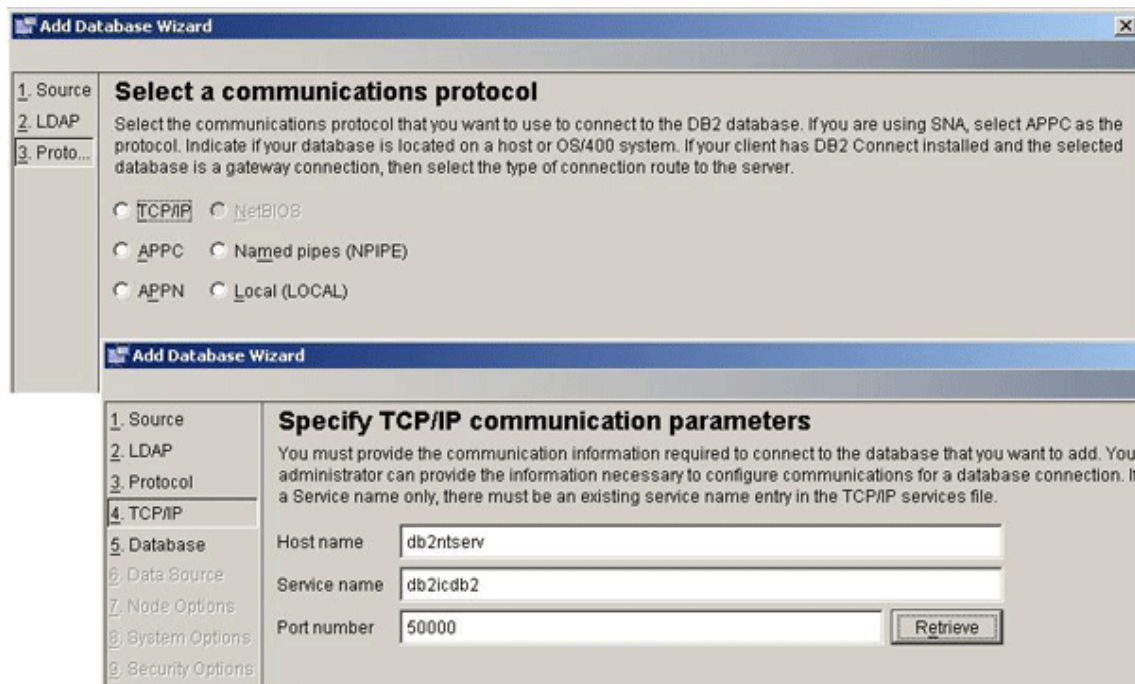
1. Use the Configuration Assistant to export information to an access profile (which is just an ASCII file). In the figure below, notice that some DB2 environment settings, such as database manager configuration and DB2 profile registries, can also be exported.

You may find it burdensome to explain to all your users how to import the profile using the Configuration Assistant. In addition, some users may have a version of the DB2 runtime client installed that does not include Configuration Assistant. In such cases, you can use the following command to perform the same import as described above:

```
db2cfimp access_profile_name
```

Configuring database connectivity manually

If you know all the information required for configuring your connectivity, you can use the Add Database wizard from the Configuration Assistant, as illustrated below:



Alternatively, you can use the `catalog` commands via the DB2 Command Line Processor (CLP) using the following steps:

1. Catalog the node.

Each instance you want to attach to needs to be cataloged as a node. Use the `catalog` command with varying keywords for each supported communication protocol. Some examples:

```
db2 catalog tcpip node mynode remote db2server.mycompany.com server db2icdb
db2 catalog netbios node jeremy remote N01FCBE3 adapter 0
```

2. Catalog the database.

Catalog one or more databases that belong to the cataloged instance. Some examples:

```
db2 catalog database sample as mysamp at node mynode
db2 catalog database baydb as newbaydb at node mynode
```

Here is a list of other catalog commands you may find useful:

- catalog appc node ...
- catalog appn node ...
- catalog ldap node ...
- catalog local node ...
- catalog named pipe node ...
- catalog netbios node ...
- catalog tcpip node ...
- catalog database ...
- catalog dcs database ...
- catalog ldap database ...
- catalog odbc data source ...

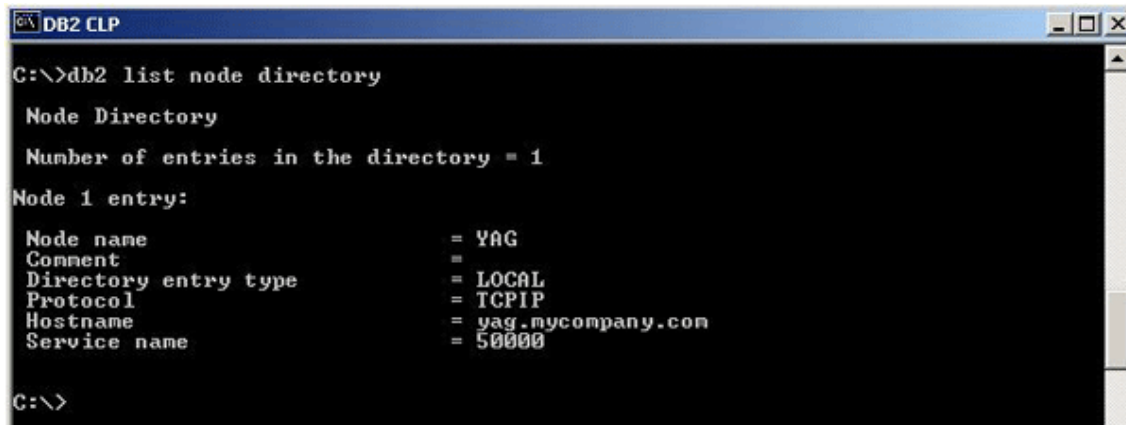
Listing node and database directories

Entries of successfully cataloged nodes and databases are stored in the `DB2 NODE` directory and in the `DATABASE` directory. They abstract away the real location of the instances and databases.

To list the node directory:

```
db2 list node directory
```

You should see output similar to this:



```
C:\>db2 list node directory

Node Directory
Number of entries in the directory = 1

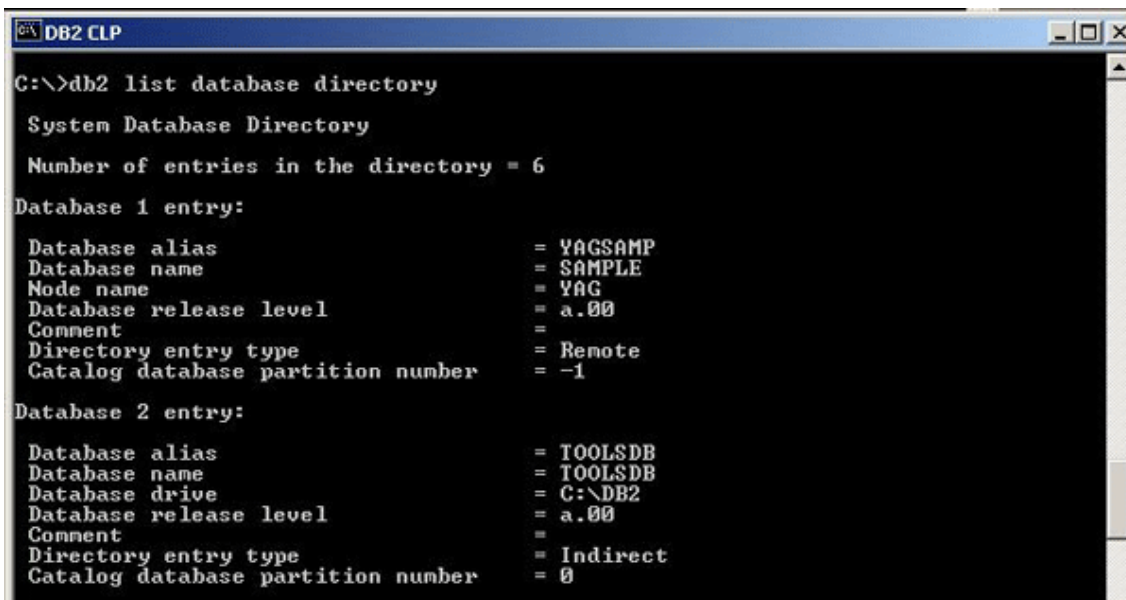
Node 1 entry:
Node name           = YAG
Comment             =
Directory entry type = LOCAL
Protocol            = TCPIP
Hostname            = yag.mycompany.com
Service name        = 50000

C:\>
```

To list the database directory:

```
db2 list database directory
```

You should see output similar to this:



```
C:\>db2 list database directory

System Database Directory
Number of entries in the directory = 6

Database 1 entry:
Database alias      = YAGSAMP
Database name       = SAMPLE
Node name           = YAG
Database release level = a.00
Comment            =
Directory entry type = Remote
Catalog database partition number = -1

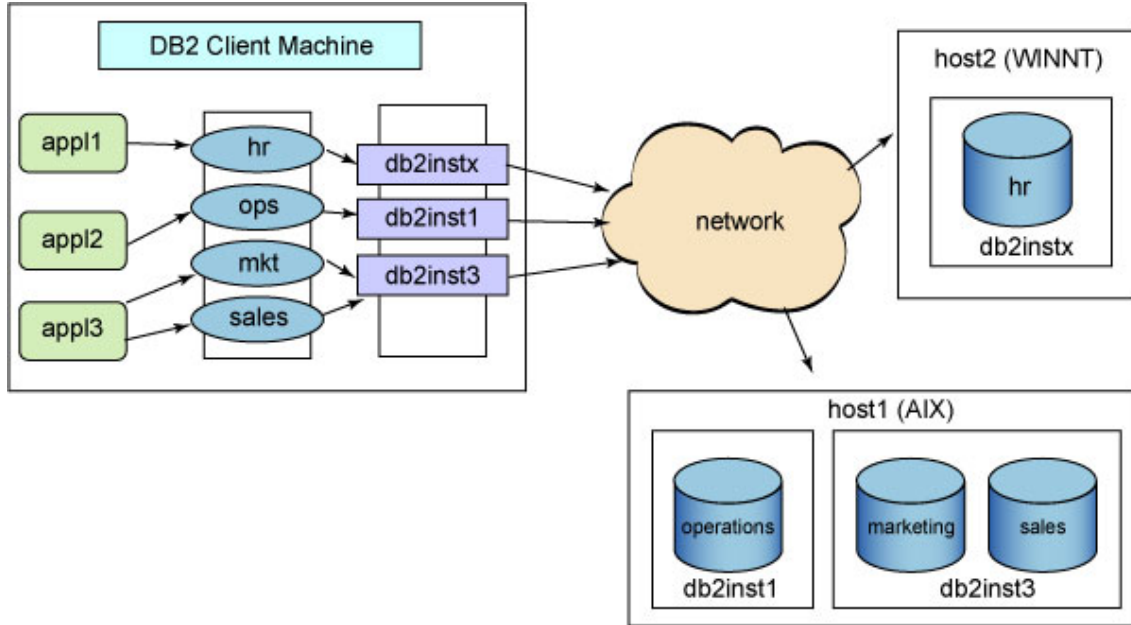
Database 2 entry:
Database alias      = TOOLSDB
Database name       = TOOLSDB
Database drive      = C:\DB2
Database release level = a.00
Comment            =
Directory entry type = Indirect
Catalog database partition number = 0
```

Example of a DB2 client/server environment

Let's use an example to get a complete picture of the DB2 client/server environment.

In the diagram below, an AIX server (host1) has two instances and three databases defined. The Windows NT server (host2) contains only one instance and one database.

For a client machine to connect to all the databases in this scenario, each remote instance must be cataloged and stored in its node directory, and each database must be pointed to its associated node (or instance).



Attaching to an instance and connecting to a database

Once you've set up client/server connectivity, you can either attach to an instance or connect to a database from the client.

To attach to an instance:

```
attach to nodename user username using password
```

nodename must be already cataloged using the `catalog` command described previously (see [Configuring database connectivity manually](#) on page 19).

With instance attachment, you can perform remote administrative tasks such as:

- Creating and dropping databases
- Retrieving, updating, and resetting database manager and database configuration parameters

- Managing database monitors
- Backing up, restoring, and rolling forward a database
- Forcing users and applications off the databases defined in the instance

To connect to a database:

```
connect to database_name user username using password
      [new new_password confirm new_password]
```

Notice that the `connect` statement also allows you to set a new password for the user specified.

With a database connection, you can manipulate data and database objects. The allowed operations are:

- Database manipulation language (DML) `SELECT`, `INSERT`, `UPDATE`, and `DELETE` statements
- Database definition language (DDL) `CREATE` database object statements
- Database control language (DCL) `GRANT` and `REVOKE` statements
- Precompiling and binding applications to the database
- Moving data using `EXPORT`, `IMPORT`, and `LOAD` commands

Disconnecting a database connection

Use the `force application` command to disconnect database connections. This command is very powerful and handy, especially when a user has trouble terminating unwanted jobs. The current transaction of the specified connection is stopped and rolled back.

First, use the `list applications` command to show all current connections made to any database defined within the instance:

```
db2 list applications [show detail]
```

Your output should look something like this:

Auth Id	Application Name	Appl. Handle	Application Id	DB Name	# of Agents
CLARALIU	db2bp.exe	7	*LOCAL.DB2.00E000150926	SAMPLE	1
CLARALIU	db2bp.exe	6	*LOCAL.DB2.00F1C0150335	SAMPLE	1
DB2ADMIN	db2dasstm.exe	5	*LOCAL.DB2.005A40134734	TOOLSDB	1
DB2ADMIN	db2dasstm.exe	4	*LOCAL.DB2.005A40134733	TOOLSDB	1

Identify the connections that you want to terminate and specify the associated application handles in the `force application` command. If multiple connections are identified, separate them with commas. The following command terminates connections associated with application handles numbered 5 and 6:

```
db2 force application (6, 5)
```

To disconnect all database connections in an instance, simply use the `all` option:

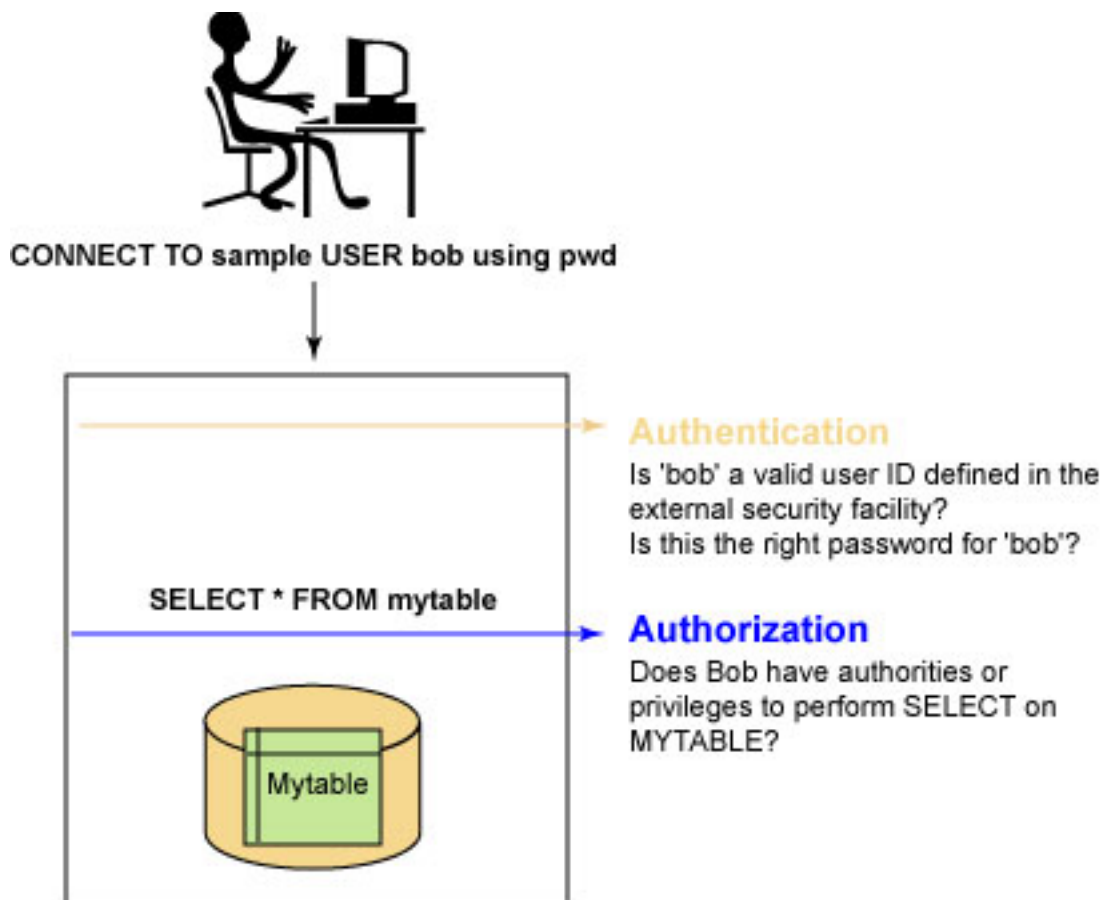
```
db2 force application all
```

The `force application` command will only terminate the connections specified. It does not stop new applications from connecting to the databases.

Section 4. DB2 security

DB2 security overview

DB2 security is handled by a combination of external security services and an internal DB2 authorization mechanism. The external security service authenticates users who want to access a DB2 server; security software outside of DB2 takes care of the authentication. This software can be a security facility of the operating system, or a separate product such as Kerberos. Once the user ID and password have been successfully verified, an internal DB2 process takes over and makes sure that the user is authorized to perform the requested operations.



Authentication types

The authentication type determines where the user ID/password pair is verified. The supported authentication types are:

- SERVER (default)
- SERVER_ENCRYPT
- KERBEROS
- KRB_SERVER_ENCRYPT
- CLIENT

We'll discuss these in detail in the following panels.

Authentication types are set at both the server and client.

At server: Only one authentication type is allowed per instance. This means the setting applies to all databases defined under that instance. Specify it in the database manager configuration file with the `AUTHENTICATION` parameter.

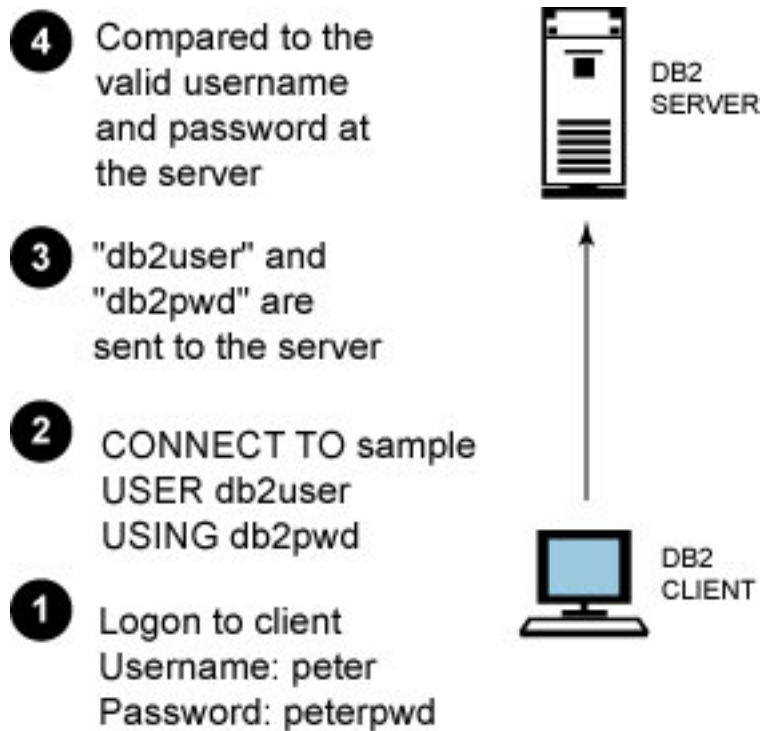
```
db2 update database manager configuration authentication auth_type
```

At client: Each database cataloged at the client has its own authentication type specified with the `catalog database` command.

```
db2 catalog database db_name at node node_name authentication auth_type
```

Authenticating with the SERVER option

With the `SERVER` option, the user ID and password are sent to the server for validation. Consider the example below:

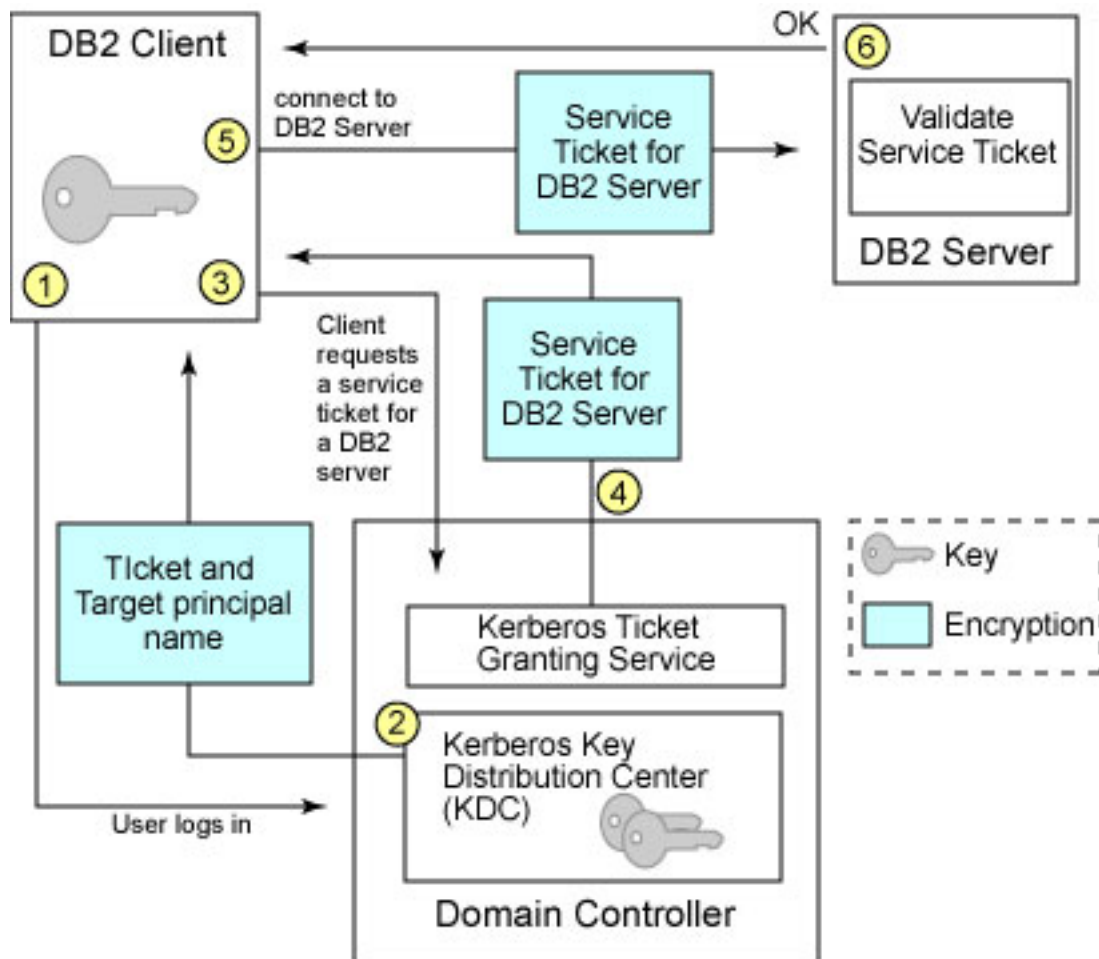


1. A user logs into a workstation with the username *peter* and password *peterpwd*.
2. *peter* then connects to the SAMPLE database with the user ID *db2user* and password *db2pwd*, which are defined at the remote DB2 server.
3. *db2user* and *db2pwd* are sent to the server through the network.
4. *db2user* and *db2pwd* are validated at the DB2 server.

If you want to protect the user ID and password from eavesdropping, use authentication type `SERVER_ENCRYPT` so that both user ID and password are encrypted.

Authenticating with Kerberos

Kerberos is an external security facility that uses conventional cryptography to create a shared encrypted key. It offers a secured authentication mechanism because the user ID and password no longer need to be transferred across the network in clear text. By using the encrypted key, it also makes single sign-on to a remote DB2 server possible. Refer to the diagram below to see how Kerberos authentication works with DB2.

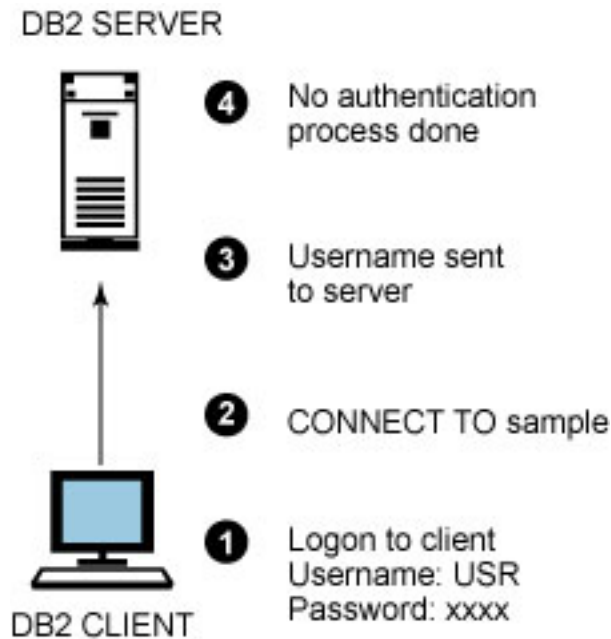


Authentication type `KERBEROS` can be used when both the DB2 client and server support the Kerberos security protocol. However, some clients may not support Kerberos, but still need to access the DB2 server. To ensure that both types of clients are able to connect securely, set the authentication type at the DB2 server as `KRB_SERVER_ENCRYPT`. This allows all Kerberos-enabled clients to authenticate with Kerberos, while other clients use `SERVER_ENCRYPT` authentication instead. The following illustration offers a quick summary of different client and server authentication settings related to Kerberos.

Client Specification	Server Specification	Client/Server Resolution
KERBEROS	KRB_SERVER_ENCRYPT	KERBEROS
Any other setting	KRB_SERVER_ENCRYPT	SERVER_ENCRYPT

Authenticating on the client

This option allows authentication to occur at the client machines. When a user successfully logs in to the client machine, a connection can be made to the database without challenging the user for a password.



It is important to understand that there are client systems that do not have a reliable security facility, such as Windows 9x and Classic Mac OS. They are referred as *untrusted clients*. Anyone who has access to these systems can also connect to the DB2 server without any authentication. Who knows what kind of destructive operations they will perform (e.g., dropping a database)? In order to provide the flexibility of allowing trusted clients to perform authentication on their own and, at the same time, forcing untrusted clients to be authenticated at the server, two other database manager configuration parameters are introduced:

- TRUST_ALLCLNTS
- TRUST_CLNTAUTH

Note that these two parameters will be evaluated only when authentication is set to `CLIENT`. We'll look at them in more detail in the next panel.

Trusting clients

`TRUST_ALLCLNTS` determines which types of clients are trusted. The parameter has the following possible values:

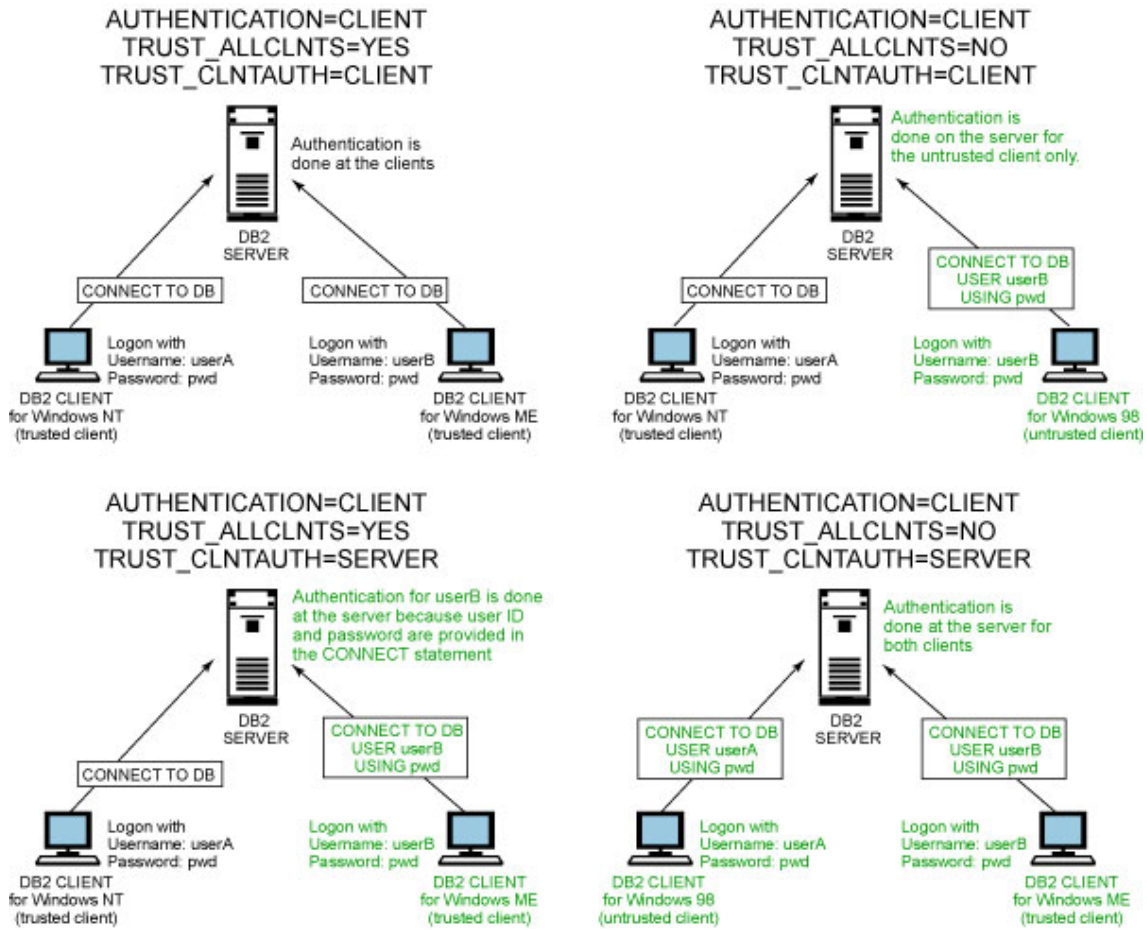
- **YES** -- Trust all clients. This is the default setting. Authentication will take place at the client. There is an exception, which we'll discuss in more detail when we cover `TRUST_CLNTAUTH` below.
- **NO** -- Trust only clients with reliable security facilities (i.e., trusted clients). For untrusted clients to connect, user ID and password must be provided for authentication to take place at the server.
- **DRDAONLY** -- Trust only clients that are running on iSeries or zSeries platforms (i.e., DRDA clients). Any other clients must provide user ID and password.

Consider a scenario in which a DB2 server has set authentication to `CLIENT` and `TRUST_ALLCLNTS` to `YES`. You log into a Windows 2000 machine as *localuser* and connect to the remote database without specifying a user ID and password. *localuser* will be the connected authorization ID at the database. What if you want to connect to the database with a different user ID -- as *poweruser*, who has the authority to perform a database backup, for instance?

To allow such behavior, use `TRUST_CLNTAUTH` to specify where authentication will take place if a user ID and password are supplied in a `connect` statement or `attach` command. Two values are allowed:

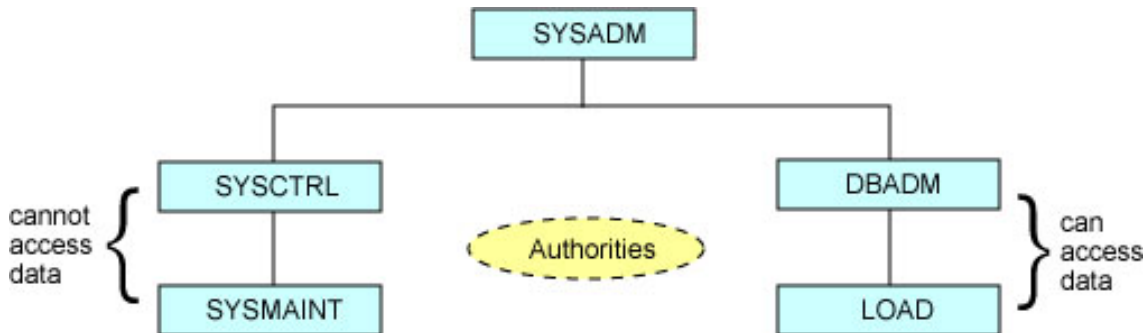
- **CLIENT** -- Authentication is performed at client; user ID and password are not required.
- **SERVER** -- Authentication is done at the server when user ID and password are supplied.

Let's look at some examples to illustrate the usage of the parameters:



Setting authority levels

Authority levels control the ability to perform database manager maintenance operations and manage database objects. There are five authorities in DB2:



- SYSADM has full privileges for managing the instance and also has access to data in the underlying databases.
- SYSCTRL and SYSMANT have certain privileges in managing the instance, its databases, and database objects. These authorities do *not* have access to the data. For example, statements such as 'SELECT * FROM mytable' or 'DELETE FROM mytable' are not allowed.
- DBADM has privileges to perform administrative tasks on the specified database. It also has full data access to the database.
- LOAD has privileges to run the load utility against the specified database.

Here is a summary of functions that each authority can perform:

Function	SYSADM	SYSCTRL	SYSMANT	DBADM	LOAD
Update DBM CFG	YES				
Grant / revoke DBADM	YES				
Establish / change SYSCTRL	YES				
Establish / change SYSMANT	YES				
Force users	YES	YES			
Create / drop database	YES	YES			
Restore to new database	YES	YES			
Update DB CFG	YES	YES	YES		
Backup database / table space	YES	YES	YES		
Restore to existing database	YES	YES	YES		
Perform roll forward recovery	YES	YES	YES		
Start / stop instance	YES	YES	YES		
Restore table space	YES	YES	YES		
Run trace	YES	YES	YES		
Obtain monitor snapshots	YES	YES	YES		
Query table space state	YES	YES	YES	YES	
Prune log history files	YES	YES	YES	YES	
Quiesce table space	YES	YES	YES	YES	
Load tables	YES			YES	YES
Set / unset check pending status	YES			YES	
Create / drop event monitors	YES			YES	

Managing DB2 authorities

The SYS* authorities are set in the database manager configuration by assigning a user group defined in the operating system or security facility to the associated parameters. It must be a group name with a maximum length of 8 characters, as illustrated in the figure below:

Database Manager Configuration

SYSADM group name	(SYSADM_GROUP) = ADM1
SYSCTRL group name	(SYSCTRL_GROUP) = CTRL1
SYSMAINT group name	(SYSMAINT_GROUP) = MAINT1

```
db2 update dbm cfg using sysadm_group adm1
db2 update dbm cfg using sysctrl_group ctr1
db2 update dbm cfg using sysmaint_group maint1
```

DBADM and LOAD are database-level authorities. They are granted to a user or a group of users with a `grant` statement and revoked with a `revoke` statement:

```
connect to sample;
grant dbadm on database to user john;
grant load on database to group dbagrp;
revoke load on database from group dbagrp;
```

Note that users with LOAD authority also require INSERT privilege on the table before data can be loaded. We'll talk more about privileges in the next panel.

Setting privileges

Privileges give users the right to access database objects in a specific way. The following lists offer a summary of privileges for different database objects.

Database privileges:

- CONNECT allows users to connect the database.
- BINDADD allows users to create new packages in the database.
- CREATETAB allows users to create new tables in the database.
- CREATE_NOT_FENCED allows users to create non-fenced user-defined functions or stored procedures.
- IMPLICIT_SCHEMA allows users to create objects in a schema that do not already exist.
- QUIESCE_CONNECT allows users to access the database while it is quiesced.
- CREATE_EXTERNAL_ROUTINE allows users to create stored procedures written in C, the Java™ language, OLE, and COBOL.

Schema privileges:

- CREATEIN allows users to create objects within the schema.
- ALTERIN allows users to alter objects within the schema.
- DROPIN allows users to drop objects within the schema.

To explicitly create a new schema, use the `create schema` command:

```
connect to sample user dbowner;  
create schema dev authorization devuser;
```

Table space privileges:

- USE OF TABLESPACE allows users to create tables within the specified table space. This privilege cannot be used on the SYSCATSPACE or any system temporary table spaces.

Table and view privileges:

- CONTROL provides the user with all privileges for a table or view as well as the ability to grant those privileges (except CONTROL) to others.
- ALTER allows users to alter a table or view.
- DELETE allows users to delete records from a table or view.
- INDEX allows users to create indexes on a table.
- INSERT allows users to insert an entry into a table or view.
- REFERENCES allows users to create and drop a foreign key, specifying the table as the parent in a relationship.
- SELECT allows users to retrieve rows from a table or view.
- UPDATE allows users to update entries in a table or view. This privilege can also limit users in updating specific columns only: `grant update (workdept, job) on table employee to devuser;`
- ALL PRIVILEGES grants all the above privileges except CONTROL on a table or view.

Package privileges:

- CONTROL provides users the ability to rebind, drop, or execute a package as well as the ability to grant these privileges (except CONTROL) to others.
- BIND allows users to rebind an existing package.
- EXECUTE allows users to execute a package.

Index privileges:

- CONTROL allows users to drop an index.

Routine privileges:

- EXECUTE allows users to execute the user-defined function.

Sequence privileges

- USAGE allows users to use NEXTVAL and PREVVAL expressions for a sequence object.

Granting explicit privileges

Granting a privilege with `grant option` allows the authorization ID to extend the specified privilege to others. This option is only available to package, routine, schema, table, table space, and view.

Although the grant privilege is extended, the revoke privilege is not. If privileges are received through the `with grant option`, a user will not be able to revoke the privileges from others. Here are some examples.

This statement allows *john* to perform `select`, `update`, or `delete` operations on the table *employee* and to grant any of these privileges to others:

```
grant select, update, delete on table employee to user john
with grant option
```

This statement allows users in the *devusers* group to rebind, drop, and execute the package *dev.pkg1*. The same group of users can also grant BIND and EXECUTE (but not CONTROL) privileges to others.

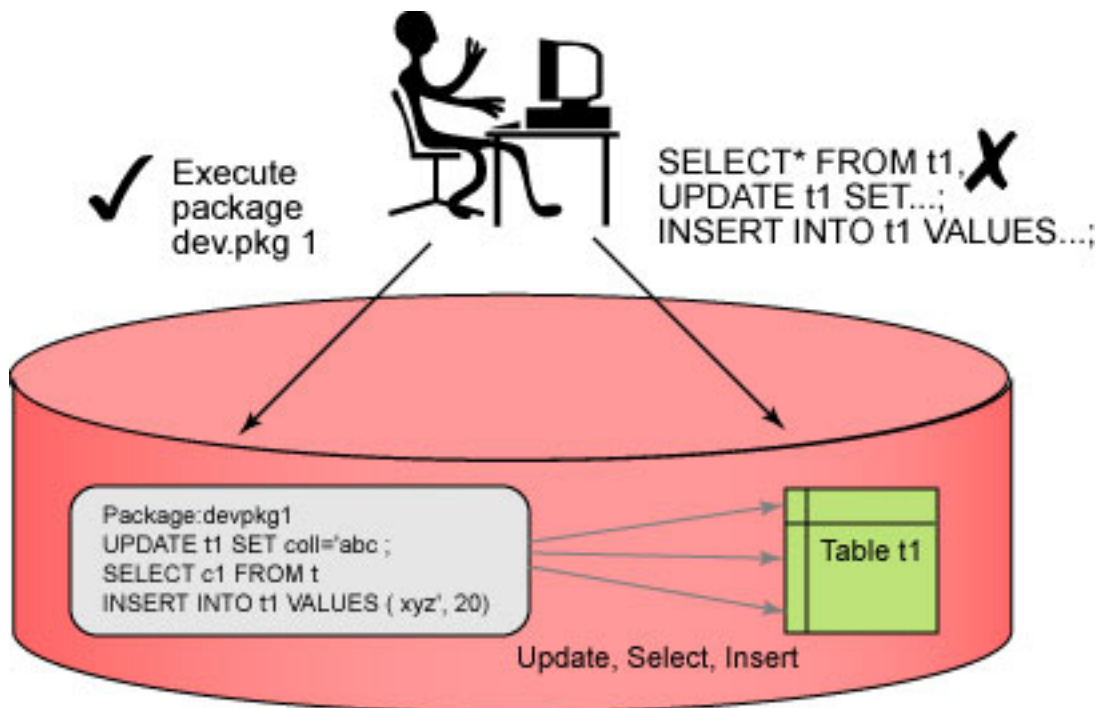
```
grant control on package dev.pkg1 to group devusers
with grant option
```

Granting implicit and indirect privileges

Typically, DB2 privileges are granted explicitly with `grant` statements as discussed previously. Sometimes users may also obtain privileges implicitly or indirectly from certain operations performed. Let's look at some scenarios.

- A user granted DBADM authority is also implicitly granted BINDADD, CONNECT, CREATETAB, CREATE_NOT_FENCED, and IMPLICIT_SCHEMA.
- When a user creates a database:

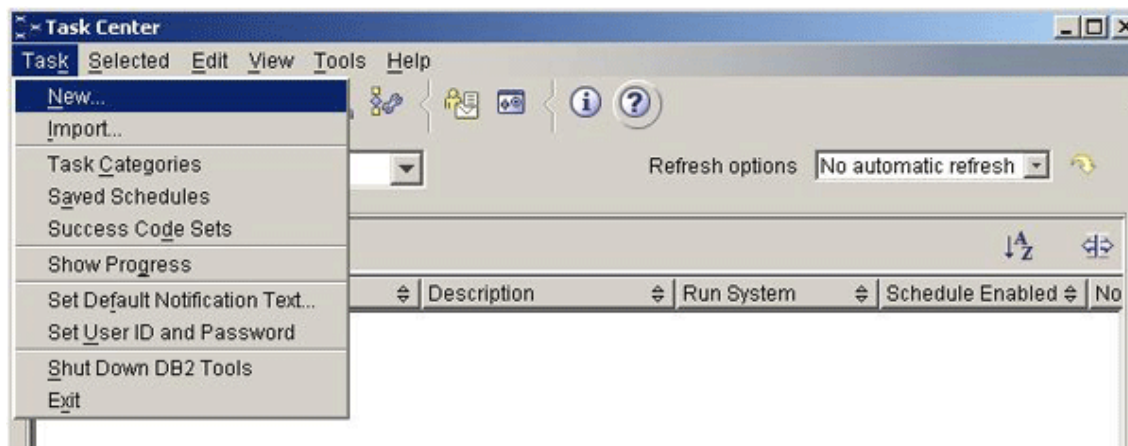
- DBADM authority is granted to the database creator.
- CONNECT, CREATETAB, BINDADD, and IMPLICIT_SCHEMA privileges are granted to PUBLIC.
- USE OF TABLESPACE privilege on the table space USERSPACE1 is granted to PUBLIC.
- BIND and EXECUTE privileges on each successfully bound utility are granted to PUBLIC.
- EXECUTE privileges with grant option on all functions in the SYSFUN schema are granted to PUBLIC.
- A user who creates a table, a view, an index, a schema, or a package automatically receives CONTROL privilege on the database object he or she creates.
- When a user executes a package that contains static SQL statements, explicit privileges for database objects referenced in the statements are not required. The user only needs EXECUTE privilege on the package to execute the statements. However, this does not mean that the user has direct access to the underlying database objects. Consider the following example:



Section 5. Scheduling jobs

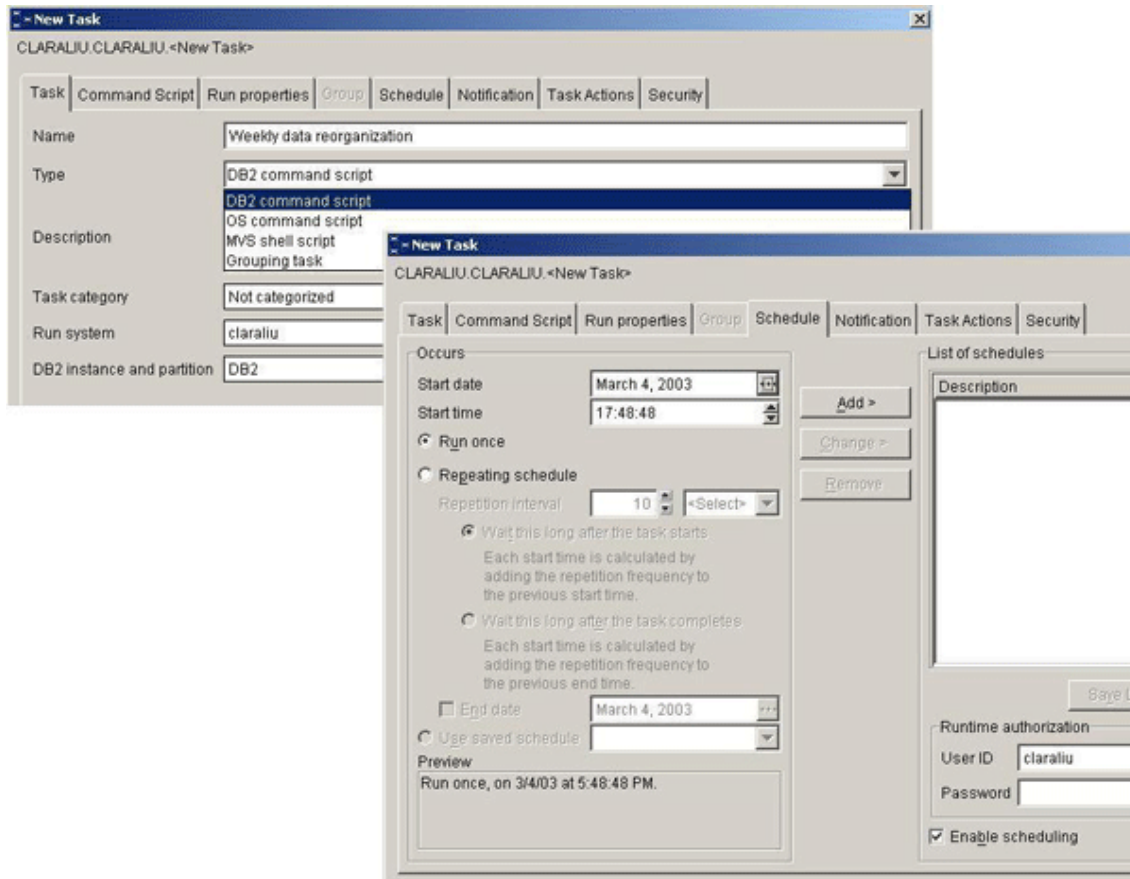
Using the Task Center

Managing a DB2 server does not just involve initial implementation of the instance and database; it also entails performing regular maintenance tasks such as REORG and RUNSTATS. DB2 has an integrated set of graphical tools to help administrators implement, manipulate, and maintain DB2 instances and databases more efficiently. The DB2 Task Center provides an easy-to-use graphical interface for organizing task flow, scheduling tasks, and distributing notifications about the status of completed tasks.



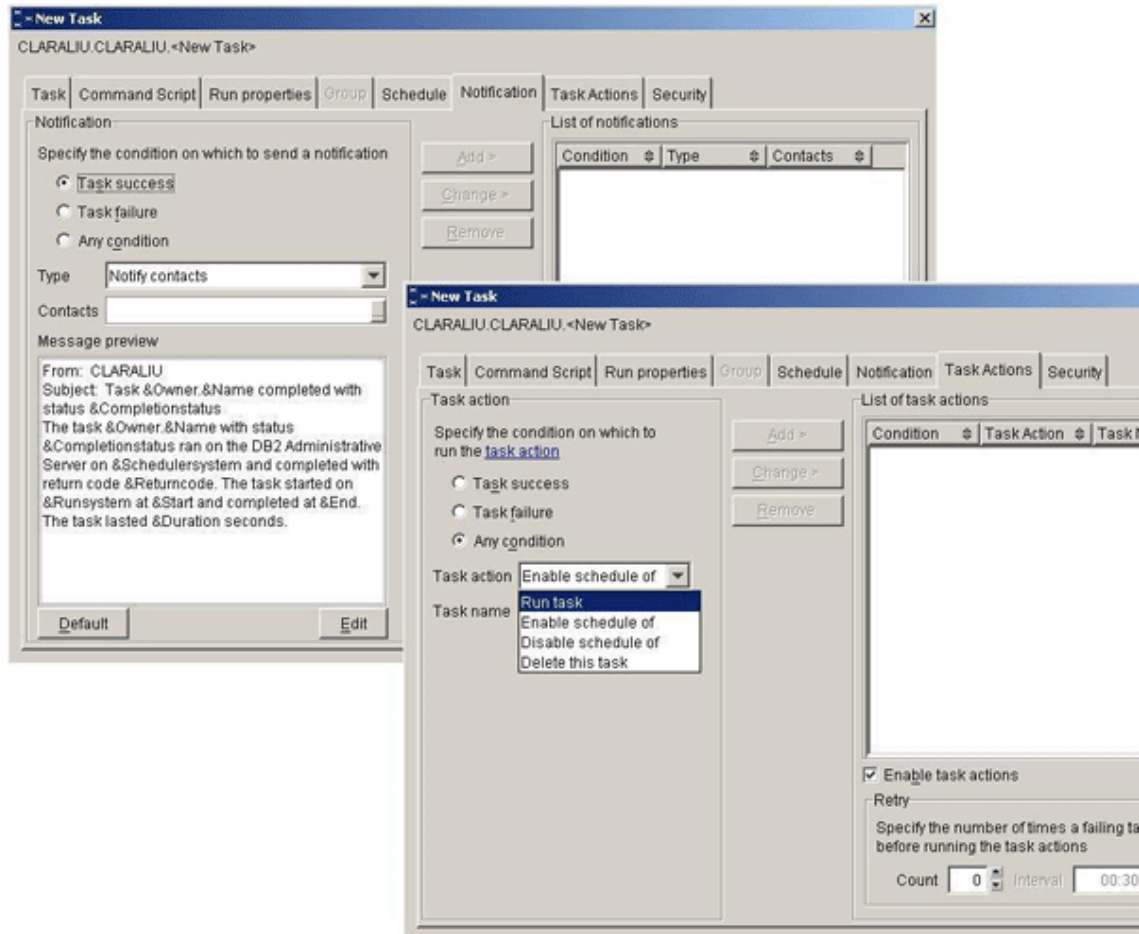
Creating a new task

The Task Center doesn't just define DB2 database scripts; it can also define OS command scripts and MVS shell scripts. To begin, enter or import a script for the task under the Command Script tab. You are then ready to schedule the task, set its frequency, and establish the authorization ID used to execute it.



Setting notifications and task actions

The Notification tab allows you to specify when to send out a notification, who to send it to, and what the notification message text sent upon completion of the task should be. In some cases, you may want to use such notifications as triggers to run other tasks, depending on the condition returned. Use the Task Action tab to run, schedule, or disable scheduling of another task.



Creating a tools catalog database

DB2 tools catalog tables and views are used to store task information created by the Task Center. They are required to enable scheduling and task automation. Tools catalog tables can be created within any existing database or in a separate database through use of the `create tools catalog` command. Regular and system temporary table space with 32 KB page sizes are required; if they are not specified in the command, they will be created by default. Here are some examples:

- The following command creates the tools catalog under the schema `toolscat` in the new database `toolsdb`:

```
db2 create tools catalog toolscat create new database toolsdb
```
- The following command creates the tools catalog under schema `toolscat` in the `tbasp32k` table space of an existing database `toolsdb`:

```
db2 create tools catalog toolscat user existing tablespace tbsp32k in database toolsdb
```

Section 6. Using notification logs

Capturing diagnostic information

DB2 uses the first failure data capture (FFDC) mechanism to automatically capture errors and warnings it encounters. Diagnostic information can be recorded in different places: the administration notification logs, DB2 diagnostic log, dump files, trap files, and (for UNIX only) core files.

As the name implies, administration notification logs are intended for use by database and system administrators. The DB2 diagnostic file (also referred as `db2diag.log`) contains detailed information for problem determination that is mainly used by DB2 customer support. The dump files capture extra information in binary format named after the failing process ID. Trap and core files are generated when DB2 terminates abnormally and cannot continue processing. These files are also binary files and could contain a memory image of the terminated process.

Understanding and knowing how to interpret the notification logs is a skill that you definitely need in order to manage a DB2 server. If you do run into problems that you cannot resolve, seek help from DB2 customer support; you'll be asked for the other log files for more detailed investigation.

Setting notification level

Each instance has one DB2 notification log called `instance_name.nfy`. On Windows NT/2000/XP platforms, the notification log is found in the system event log and can be reviewed through the Windows Event Viewer. On UNIX, `instance_name.nfy` is found in the home directory of the instance owner under `sqliib/db2dump`.

Information recorded in the administration logs can be written by DB2, the Health Monitor, and user applications. The NOTIFYLEVEL database manager configuration parameter determines what level of information will be captured. There are five levels of information possible:

- 0: No administration notification messages captured. This setting is not recommended.
- 1: Only fatal or unrecoverable errors are logged.
- 2: Conditions are logged that require immediate attention. This level also captures Health Monitor alarms.
- 3: This is the default setting. It captures Health Monitor alarms, Health Monitor warnings, and Health Monitor attentions.

- 4: Only informational messages are captured.

Note that DB2 captures all the levels of information up to and including the value set in NOTIFYLEVEL. For example, if NOTIFYLEVEL is set to 3, information of levels 1, 2, and 3 is logged.

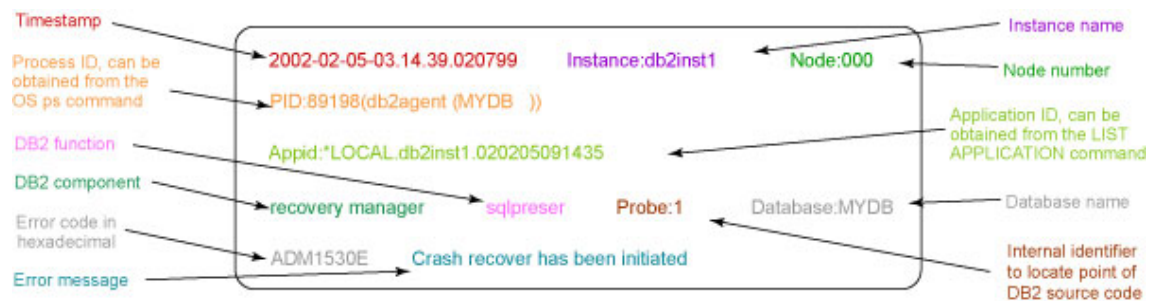
Interpreting the DB2 notification log

DB2 appends new errors, warnings, or informational events to the end of the DB2 notification log. Note that the administration log grows continuously. One good use of the Task Center would be to back up and then delete (or simply rename) these logs. Schedule such a task to run regularly.

Each event log entry is made up of different parts:

- A timestamp indicating when the event occurs
- The location reporting the message, such as the instance name, node ID, database name, process ID, application ID, or name of the DB2 or user application function
- The error type and number in hexadecimal code; this usually starts with DIA or ADM
- A diagnostic message explaining the error

The figure below illustrates a typical log entry:



Section 7. Conclusion

Summary

This tutorial introduced you to the basic knowledge and skills of managing DB2 servers. Every server has one or more DB2 instances. An instance provides a logical environment in which DB2 commands and functions can execute. In the first part of the tutorial, you learned how to create, drop, start, stop, list, migrate, and update instances. The DB2 environment plays an important role in influencing how DB2 behaves. It is made up of the operating system environment settings, DB2 profile registries, database manager, and database configuration parameters. They can be easily configured with commands as illustrated in the tutorial.

You learned three methods of configuring DB2 client and server connectivity. You can search the network with DB2 discovery, use DB2 access profiles, or specify communication information manually. The Configuration Assistant is available to use any of the methods to set up the node and database catalog directories. With proper connectivity configuration, you can attach to an instance to perform remote administration tasks and connect to a database for data access.

Managing a DB2 server also includes managing its access. DB2 security is composed of authentication and authorization. Authentication is performed externally by a security facility. There are different authentication types that allow you to control where authentication should take place. Once a user is authenticated, DB2 will check to make sure the user is allowed to perform the requested operations. Different levels of authorities and privileges are available to support granular security control.

You were also introduced to task creation and automation. Use the Task Center to create tasks that are coded in DB2 commands, OS commands, or MVS shell commands. The tool has options to enter instructions for notification purposes and also specify other tasks to perform upon completion.

The DB2 administration notification log records errors and warnings raised by DB2, the Health Monitor, and user applications. Each event log entry contains information such as timestamp, database name, application ID, DB2 function and component that raise the message. This is definitely a good place to start in troubleshooting errors encountered in DB2.

Resources

Check out the other parts of the DB2 V8.1 Database Administration certification prep series:

- [Data Placement: DB2 V8.1 Database Administration certification prep, Part 2 of 6](#)
- [Database Access: DB2 V8.1 Database Administration certification prep, Part 3 of 6](#)
- [Monitoring DB2 Activity: DB2 V8.1 Database Administration certification prep, Part 4 of 6](#)
- [DB2 Utilities: DB2 V8.1 Database Administration certification prep, Part 5 of 6](#)
- [Backup and Recovery: DB2 V8.1 Database Administration certification prep, Part 6 of 6](#)

You can learn more about managing DB2 from the following resources:

- [DB2 Version 8 Administration Guide: Performance](#)
 - Chapter 13. Configuring DB2
- [DB2 Version 8 Installation and Configuration Supplement](#)
 - Chapter 2. Setting up the DB2 server after manual installation
 - Chapter 3. Configuring client to server communications
 - Chapter 4. Configuring DB2 server communications
- [DB2 Version 8 Administration Guide: Implementation](#)
 - Chapter 1. Before Creating a Database
 - Chapter 4. Controlling Database Access

For more information on the DB2 V8.1 for Linux, UNIX, and Windows Database Administration Certification (Exam 701), check out these links:

- [IBM Data Management Skills information](#)
- Download a [self-study course for experienced Database Administrators \(DBAs\)](#) to quickly and easily gain skills in DB2 UDB.
- Download a [self study course for experienced relational database programmers](#) who'd like to know more about DB2.
- [General Certification information](#) -- including some book suggestions, exam objectives, and courses.
- Check out the [IBM Certification Exam Tool \(ICE\)](#), where you can use pre-assessment/sample tests to prepare for exams leading toward IBM certification.
- Check out [developerWorks Toolbox](#) for one-stop access to over 1,000 IBM tools, middleware, and technologies from DB2, Lotus, Tivoli, and WebSphere for open standards-based Web services and application development.

Feedback

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The open source Toot-O-Matic tool is an XSLT style sheet and several XSLT extension functions that convert an XML file into a number of HTML pages, a zip file, JPEG heading graphics, and two PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML. (It also saves our production team a great deal of time and effort.)

You can get the source code for the Toot-O-Matic at www6.software.ibm.com/dl/devworks/dw-tootomatic-p. The tutorial [Building tutorials with the Toot-O-Matic](#) demonstrates how to use the Toot-O-Matic to create your own tutorials. developerWorks also hosts a forum devoted to the Toot-O-Matic; it's available at www-105.ibm.com/developerworks/xml_df.nsf/AllViewTemplate?OpenForm&RestrictToCategory=11. We'd love to know what you think about the tool.