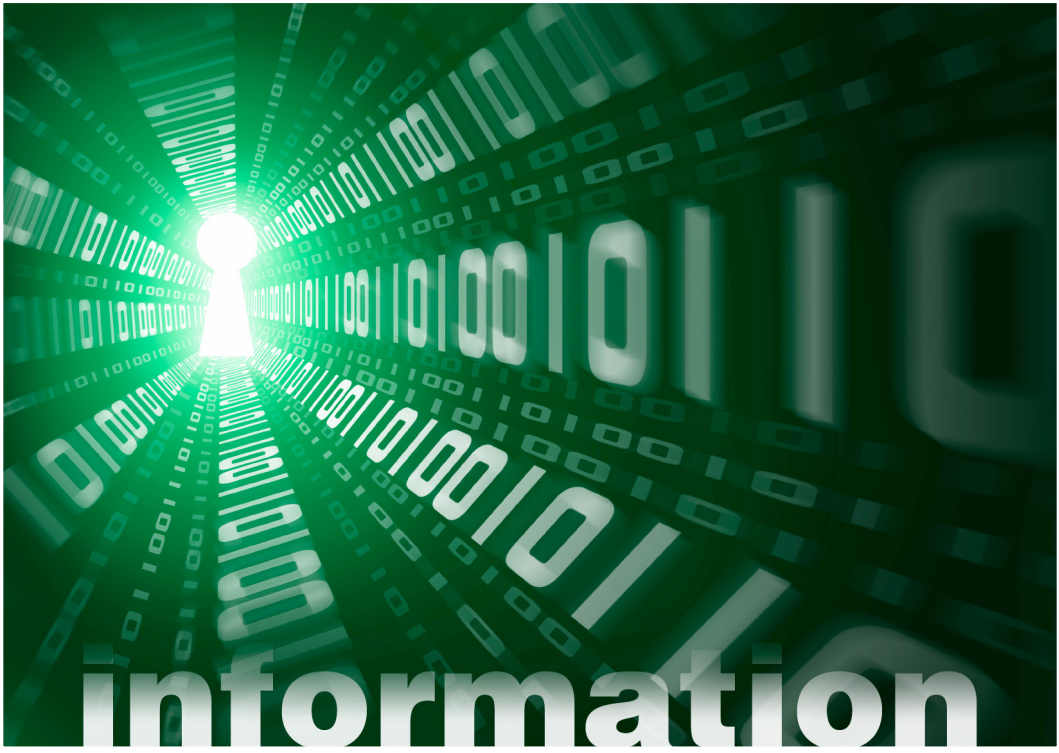
IBM

Informix. software

Informix Dynamic Server Application Development: Getting Started



Jacques Roy
Information Management Technical Product Manager
IBM Software Group

Learn how to start developing state of the art applications using Informix Dynamic Server (IDS) as the database. It includes an introduction to IDS, its installation, the installation of the client drivers, and basic information for using IDS with Java, PHP, Ruby on Rails, and.Net.

Informix Dynamic Server Application Development: Getting Started

Table of contents

1

Introduction to IDS

It is possible to find introductions to Informix® Dynamic Server (IDS) in many books and articles. This introduction is different because it focuses on what a developer should know about IDS. It assumes basic knowledge of relational database systems and the SQL language.

IDS Architecture

IDS is based on a multi-threaded, client-server architecture. This means that an IDS database server can support a large number of client connections that request access to data. Figure 1-1 illustrates the architecture.

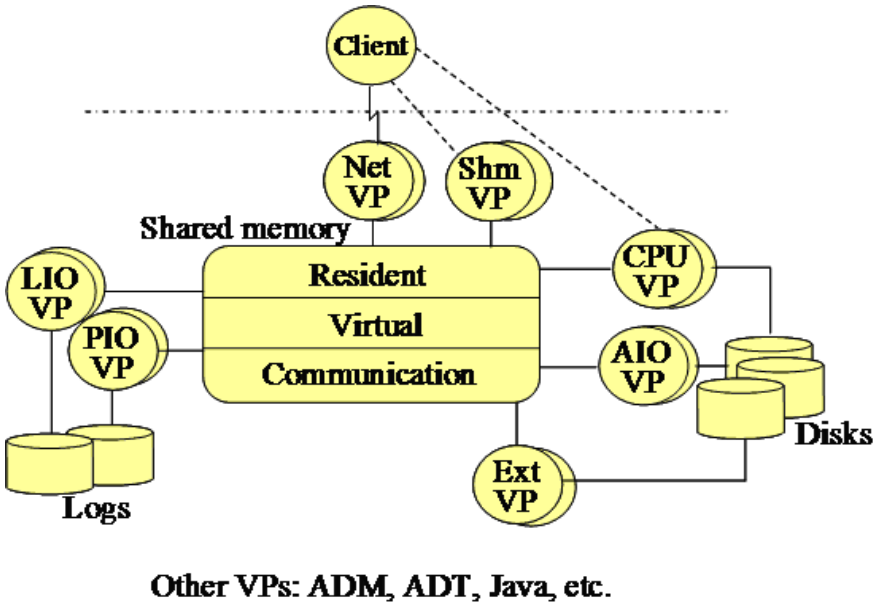


Figure 1: IDS Architecture Diagram

The IDS database server environment includes shared memory, disk storage and virtual processors (VPs). There are several type of VPs that execute specific tasks such as general SQL queries processing (CPU), physical and logical log processing (PIO and LIO), network connection processing (Net), and so on. These virtual processors are represented as processes in UNIX-type systems, and threads in the Windows® environment. This means that in UNIX®-type systems you will see a number of processes starting where in Windows, you'll only see one process. In all cases, the program executed is called `oninit`.

IDS's virtual processors communicate between them using shared memory structures. Of course, shared memory is also used to buffer database data to speed up processing.

User Connections

A user can connect to the database server using different types of connections, the most common one being a network connection (TCP). Other types of connections include shared memory and pipes.

Depending on how IDS is set up and the type of connection used, the user connection goes through a CPU VP, a NET VP, or a SHM VP. This is transparent to the user.

You should always keep in mind that any type of connection to the database server includes a certain level of overhead. For this reason, you want to make sure you optimize the use of your connections. This means that you do not want to open and close database connections. Instead, you should re-use them if possible.

Authentication

The connection to the database server also includes the authentication of the user. The default authentication is based on the existence of that specified username in the operating system environment. It is possible to use other types of authentications such as using a lightweight directory access protocol (LDAP) directory or single sign-on through Kerberos.

User Requests Processing

When IDS receives user SQL statements (and procedure calls), it needs to parse the statement to validate the syntax, figure out the possible execution plans and pick the best plan for execution. This section of the execution can be minimized when a statement is executed multiple times by using what is called prepared statements. IDS can take it one step further by sharing the prepared statements between users. See the IDS documentation for more details.

A statement execution thread is scheduled to run on a CPU VP. The number of CPU VPs usually maps to the number of processors on the physical machine. This way, multiple statements can be executed simultaneously.

When a statement executes, it received a quantum of time for execution. The statement can be re-scheduled if it asks for blocking operations such as disk I/O or if it uses its entire time quantum. It is then put back in the execution queue and gets rescheduled for execution as it comes to the head of the queue. This approach gives execution time to all the executing statements, allowing them to make progress in their execution no matter how many statements are running. The overall result is smoother execution and less system management requirements. This approach is similar to what operating systems do to give the illusion that each user has the machine to herself.

IDS Administration

IDS has a reputation for low administration requirements. It is often said: “set it and forget it”. To support this, IDS includes multiple autonomic capabilities that tune the database server based on the current workload. As a developer, you should at least know about log space, the SQL history feature and the Open Admin Tool for IDS (OAT).

An SQL statement executes in the context of a transaction. To insure the integrity of the data in the database, IDS uses logging to be able to restore the database to what it was before the transaction

executed if a transaction is rolled back. Some transactions, such as loading a large number of rows in a table, could cause the system to run out of log space. If you run out of log space, the system will hang. You can easily add more log space by using the OAT utility. In the case of loading a large number of rows in a table, you can change the table type to avoid logging and change it back after the load. For more information, consult the "`CREATE TABLE`" and "`ALTER TABLE`" SQL statements in the SQL syntax manual.

When developing applications, IDS can track the execution of your SQL statements. This way, you can easily identify which statement used the most time and figure out how to make it run faster by adding indexes or rewriting it. This also can be done through the OAT interface.

OAT is a web-based administration tool written in PHP. This tool helps you keep track of the health of your system. Its installation is covered in Chapter 5 on setting up a PHP environment.

Additional basic administration information is discussed in Appendix A.

IDS Capabilities

IDS is at its core an object-relational database system (ORDBMS). This means that it is a relational database system (RDBMS) with the ability to adapt to the environment by adding business types and processing to it. The rest of this chapter discusses the major capabilities that all application developers should know about. This should not be considered an exhaustive list of features, so consult the documentation for more details. Knowing about these capabilities may change the way you will implement your solutions.

First, IDS is a fully relational database system that includes all the expected features such as DDL (`CREATE`, `DROP`, and so on) and `DML` (`SELECT`, `INSERT`, `UPDATE`, `DELETE`) statements. It also includes support for features such as referential integrity, triggers, and stored procedures. Note that IDS also supports triggers on `SELECT` statements.

Sysdbopen and Sysdbclose Procedures

IDS provides the ability to create a procedure that automatically executes when a user opens a connection to the database. This can be used to set connection attributes independently from the connecting application. It can also be used to execute code such as recording who connected to the database, when. Consider the following code:

```
CREATE PROCEDURE public.sysdbopen()
  SET ISOLATION TO REPEATABLE READ;
  INSERT INTO AccessLogTable
    VALUES(USER, "O", CURRENT::DATETIME YEAR TO SECOND);
END PROCEDURE;
```

The procedure does two things: it sets the concurrency policy for the connection and records the opening of the database by that user in the `AccessLogTable` table.

The `sysdbopen()` procedure does not take any argument and does not return a value. Note that the procedure name includes a prefix. In our example, the prefix is `public`. In this case, it means that any user that does not have a `sysdbopen()` defined for their username will execute the public procedure when they open a connection to the database. This means that you can have a multiple `sysdbopen()` procedures defined, up to one procedure per user, and a "public" procedure that, applied to any user,

does not have a specific `sysdbopen()` procedure.

Similarly, the `sysdbclose()` procedure executes automatically when the connection to the database is closed. It supports up to one procedure per user and a public version for everyone.

User-Defined Types

A relational database server includes a set of data types such as integer, float, and money. IDS added types such as `ROW`, `SET`, `MULTISET`, and `LIST` to accommodate more object-oriented approaches. Other database vendors added their own data types based on some needs they saw in the marketplace.

IDS provides the ability to create new data types either based on existing ones (distinct types) or created from scratch (opaque types).

A distinct type can be useful to implement strong typing in the database server. For example, if you want to manage multiple currencies in the database, the MONEY type is not sufficient since it does not completely describe the type used and it would not return an error when two different currencies are used together such as adding yen to dollars. The following code shows the creation of a new type dollar based on money:

```
CREATE DISTINCT TYPE dollar AS MONEY;
```

This new type would not allow you to add `money` to `dollar` without an explicit cast.

An opaque type is built from scratch where you define its internal and external representations. This gives the flexibility to implement any business types that are required in a specific environment. A simple example would be the implementation of globally unique identifier (GUID). The internal representation is 16 bytes in length and the external representation is 36 bytes. One benefit would be in the saving in storage since the internal representation is only 45% of the external representation. Other opaque types could be used to better represent a hierarchical relationship between business objects such as in a bill-of-material application. We discuss this briefly in the DataBlade® section below.

User-Defined Routines

User-defined types go hand-in-hand with user-defined routines (also referred to as user-defined functions). They allow for the implementation of casting functions, conversion functions and for the implementation of business rules. Depending on the context, these functions can be written in “C”, Java, or in the Informix stored procedure language (SPL).

Let’s take date manipulation, for example. A company may need to look at activities based on quarters (3-month periods). There can be multiple definitions of quarters:

- Calendar quarter: starting on January 1st of the year
- Business quarter: starting at an arbitrary point in the year. For example April 1st
- School year quarter: starting on September 1st

Some companies require the use of multiple quarter definitions in different parts of their processing. The following is a definition of the quarter function based on the calendar year:

```
CREATE FUNCTION quarter(dt date)
RETURNS integer
WITH (NOT VARIANT)
```

```
RETURN (YEAR(dt) * 100) + 1 + (MONTH(dt) - 1) / 3;
END FUNCTION;
```

This function is written in SPL and takes advantage of built-in functions `YEAR` and `MONTH`. The "`WITH(NOT VARIANT)`" statement indicates that the function will always return the same value given a specific input value (date) and also indicates that it does not modify the content of the database. This allows for additional optimization by IDS.

The `quarter()` function takes a date as input and returns an integer as a result. The format of the result value is `YYYYqq` where `YYYY` represents the year and `qq` is a 2-digit representation of the quarter (01-04). So the fourth quarter of 2008 would be 200804. This implementation represents one choice. For example, we could have decided to return a character string instead.

The addition of this function allows us to tell IDS how to sort and group our data based on quarter. Consider the following statement:

```
SELECT quarter(order_date), SUM(amount)
FROM orders
WHERE quarter(order_date) BETWEEN 200801 AND 200804
GROUP BY 1
ORDER BY 1;
```

This statement returns four values representing the income for each quarter of the year 2008. IDS `is able` to figure out which date belongs to which quarter and sum up the amount to give us the desired result. We can take it one step further. IDS allows for the creation of indexes on the result of a function. We can create the following index:

```
CREATE INDEX orders_quarter_idx
ON orders( quarter(order_date) );
```

The SELECT statement above could take advantage of the new index. If we had 10 years of data in the table, this index would improve performance by eliminating I/O operations.

A quarter function could also be created to process according to a business year starting on April 1st. This means that April 1, 2008 is the first day of the first quarter of 2009. The following function implements our business quarter definition:

```
CREATE FUNCTION bizquarter(dt date)
RETURNS integer
WITH(NOT VARIANT)

DEFINE yr int;
DEFINE mm int;

LET yr = YEAR(dt);
LET mm = MONTH(dt) + 9; -- Apr. to Jan. is 9 months
IF mm > 12 THEN
  LET yr = yr + 1;
  LET mm = mm - 12;
END IF

RETURN (yr * 100) + 1 + (mm - 1) / 3;

END FUNCTION;
```

This function is similar to the `quarter()` above but does slightly more complex date processing to

accommodate the different start date.

For more discussion on date processing, see the “Date Processing in IDS” article listed in the reference material section.

This example gives us a glimpse of what can be done with user-defined routines. In general, if we have to process a large amount of data to get to a simple answer, it is better to use user-defined routines than transferring all the data to the application for processing.

User-Defined Aggregates

IDS provides multiple aggregate functions. They are: `AVG`, `COUNT`, `MAX`, `MIN`, `RANGE`, `STDEV`, `SUM`, and `VARIANCE`. Business processing may include many additional aggregations. This could force the transfer of large amount of data to the application instead of doing the aggregate in the database server.

IDS supports the creation of user-defined aggregates. The overall benefits include faster performance and simpler code. Here is a simple example. A bank includes a number of branches and each branch gives out loans. The bank may want to calculate the amount of risk taken by each branch to find out if some are outside acceptable risks. It is possible to create an average risk user-defined aggregate `(AVGRISK)` to take care of the calculation for us. Finding out out-of-compliance branches would then be reduced to:

```
SELECT branch_id, AVGRISK(loans)
FROM loans
GROUP BY 1
HAVING AVGRISK(loans) > 1.0
ORDER BY 2 DESC;
```

This statement expresses the following: Take all the loans for each branch and calculate the average risk. Return the value if the average risk is greater than 1.0 (arbitrary value) and return the result ordered by amount of risk taken in a descending order.

We passed `loans` to the `AVGRISK` aggregate which is the name of the table. This means that instead of passing a value such as an integer to the `AVGRISK` aggregate, we pass the entire row from the table loans. This shows a difference between the built-in aggregate functions and user-defined aggregates functions, the latter having additional flexibility.

Virtual Index/Table Interface

There are more advanced capabilities that allow for the implementation of access methods. This means that we can make something look like a table or an index to IDS. This is used in several DataBlades and in the “Informix flat-file Access” article listed in the reference material section.

This is a more advanced functionality that could be compared to operating systems’ device drivers.

DataBlades

The term DataBlade has been used multiple times in this chapter. A DataBlade, or DataBlade module, is a packaging of functionality that can be added to a database to extend the capabilities of the server to better support a specific problem domain. IDS includes utilities to register and de-register DataBlade to make it easy to administer. IDS includes several DataBlades that can make it easier to solve your

problems. They include:

- Basic Text Search (BTS): This DataBlade provides text search capabilities. It allows for word and phrase searches including wildcard searches, proximity and fuzzy searches.
- Binary data type: This DataBlade defines two new indexable binary types. It also includes several functions to operate on these types.
- Hierarchical Data Type: Provides a type and related functions to manipulate hierarchies.
- Large Object Locator: Provides a consistent interface to large objects stored inside or outside the database.
- MQ DataBlade: This provides an interface to WebSphere MW products.
- Spatial DataBlade: This DataBlade adds new spatial data types such as point, line and polygon as well as the capability to index them. This DataBlade does not come with IDS but can be downloaded for free separately on the supported platforms.
- Web Feature Services: Lets you add an Open GeoSpatial Consortium (OGC) web feature service as a presentation layer for the spatial and geospatial DataBlade modules.

There are additional DataBlade modules available at an additional cost. They include Excalibur Text Search, Geodetic, TimeSeries, TimeSeries Real-Time Loader, and more.

Reference Material

- *IBM Informix Guide to SQL: Syntax, Version 11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-7751-01
- *Using Globally Unique Identifier with IDS 9.x*
  http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0401roy/index.html
- *Date Processing in IDS*
  http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0510roy/
- *Informix Flat-File Access*
  http://www-128.ibm.com/developerworks/db2/zones/informix/library/demo/ids_ffvti.html
- *IBM Informix Database Extensions User's Guide, v11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-9427-00
- *IBM Informix Spatial User's Guide, v8.21*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=G229-6405-01
- *IBM Informix Dynamic Server Administrator's Guide, v11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-7748-01
- *Administering Informix Dynamic Server,* Carlton Doe, Sept 2008,
  ISBN 978-1-58347-076-3

2

Installing IDS

This chapter explains how to install Informix Dynamic Server (IDS) on your system. The goal is to keep it as simple as possible instead of going through all the possibilities of the installation such as choosing which component to install and the use of role separation. Once you are ready to install a production system, you can consult the documentation listed at the end of the chapter to see what other options you can take advantage of.

To be ready to use IDS for application development you need to:

- Install the product
- Set up environment variables
- Set up automatic starting and shutting down the database server
- Add storage to the database server
- Create a database

We go through all these steps in this chapter.

IBM recommends that your system has 600MB of disk space and 256MB of memory for Windows, 750MB of disk space and 256MB of memory for UNIX and Linux, and 750MB of disk space and 512MB of memory for MAC OS X. Of course, applications requirements may increase memory and disk usage.

IDS Virtual Appliance

An alternative of installing IDS and all the related drivers is to use the IDS virtual appliance. The appliance consists of a virtual image (vmware) that consists of the SUSE Linux Enterprise 10 SP2. It includes the following software:

- Informix Dynamic Server 11.50.xC4
- Informix Client SDK 3.50.xC4
- Informix JDBC Driver 3.50
- Informix Spatial DataBlade 8.21
- Informix Web DataBlade 4.13
- Data Server Driver for JDBC/SQLJ
- Data Server Driver for ODBC/CLI
- OpenAdmin Tool for IDS 2.24
- Data Studio Developer 2.1
- AGS Server Studio
- GNU C/C++ compiler and gdb debugger
- IBM JDK v1.5.0

The virtual appliance is also available in VMWare ESX, and Amazon Web Service AMI formats. You can obtain the virtual appliance image at the following locations:

- Direct Download link
  https://www14.software.ibm.com/webapp/iwm/web/reg/download.do?source=swg-informixfpd&S_PKG=dl
  It requires an IBM Internet ID and Password.

- VMware Virtual Appliance Marketplace
  http://www.vmware.com/appliances/directory/90643
- IBM Virtual Appliances Portal
  http://www.ibm.com/developerworks/wikis/display/im/IBM+Virtual+Appliances#IBMVirtualAppliances-ids
  This site includes the readme and installation instructions.

Considering the content of the appliance, it may make sense for you to use it and tailor it to your development needs instead of going through the IDS installation and related products.

Getting IDS

There are several editions of IDS available: Enterprise, Workgroup, Express, and developer. The Developer edition is a free version of IDS to allow everyone to develop and test their applications without having to pay for the database server. This is the version we are referring to in this chapter.

If you already have IDS 11.5 Developer Edition, you can safely move on to the next section.

You can download IDS 11.5 Developer Edition from the following Web page:

`http://www.ibm.com/developerworks/downloads/im/ids/learn.html?S_CMP=rnav`

You will eventually be asked to register. If you don't have an account on the IBM site, simply create it. There are no charges to register with IBM.

The download file is a 396MB zip file for Windows, a 293MB .tar file for Linux 32-bit and a 220MB .dmg file for Mac OS X.

Installing IDS

The first step of the installation is to extract the product files (iif.11.50*).

- On UNIX and Linux: `tar xvf <filename>`
- On Windows: You should be able to click on the zip file to extract its content.
- On MAC OS X: open the self-extracting `.dmg` file.

The installation procedure is similar on all platforms. Note that you will need to have superuser privileges (root) on UNIX and Linux. On MAC OS X, you will have to enter the administrator password.

Do the following to start the installation:

On UNIX and Linux: `./ids_install -gui`

On Windows: Click on `launch.exe`. If absent, click on `setup.exe`

On MAC OS X: Opening the installation file should start the installation. You will have to enter the administrator password.

The GUI installation process is mostly identical on all platforms. In the following discussion, we point out the important points and some differences between platforms. Not all screens are shown since it does not

add much to the understanding of the installation but all the steps are discussed.

On Windows, the first screen to appear is shown in figure 1.

Figure 1: Windows IDS installation welcome screen

To move to the installation proper, you have to click on “Install Products”. It is followed by an installation welcome where you simply have to click Next. On the other platforms, you have to go through a few screens such as welcome, software license agreement and installation directory before getting to the product selection. Just pick the default values.

The following screen provides the products selection to install. The Windows screen is shown in figure 2.

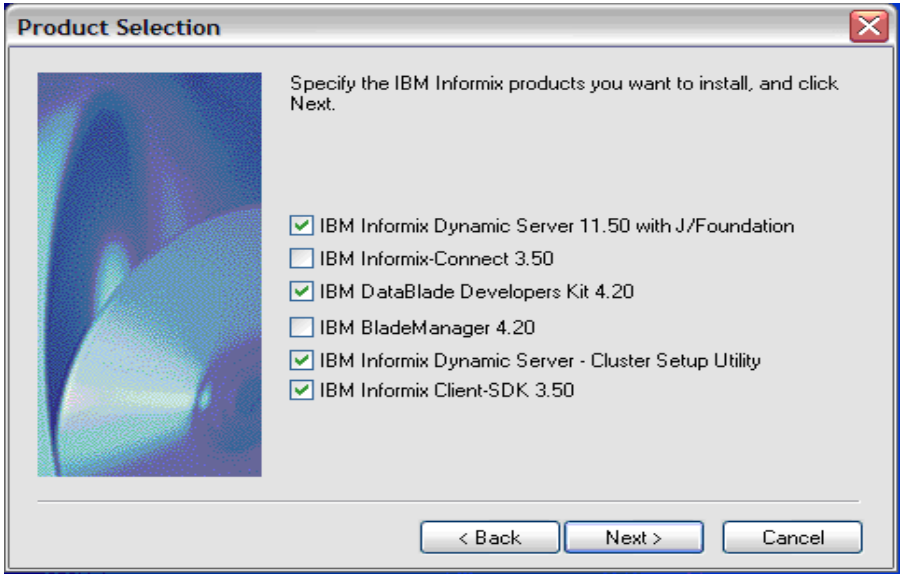


Figure 2: IDS on Windows Product Selection

Before you click Next, please de-select “IBM DataBlade Developers Kit 4.20” and “IBM Informix Dynamic Server – Cluster Setup Utility”.

The Linux Product Selection screen is shown figure 3. In this case, the default product selection is what we want so we simply click Next. The same should apply for MAC OS X installation.

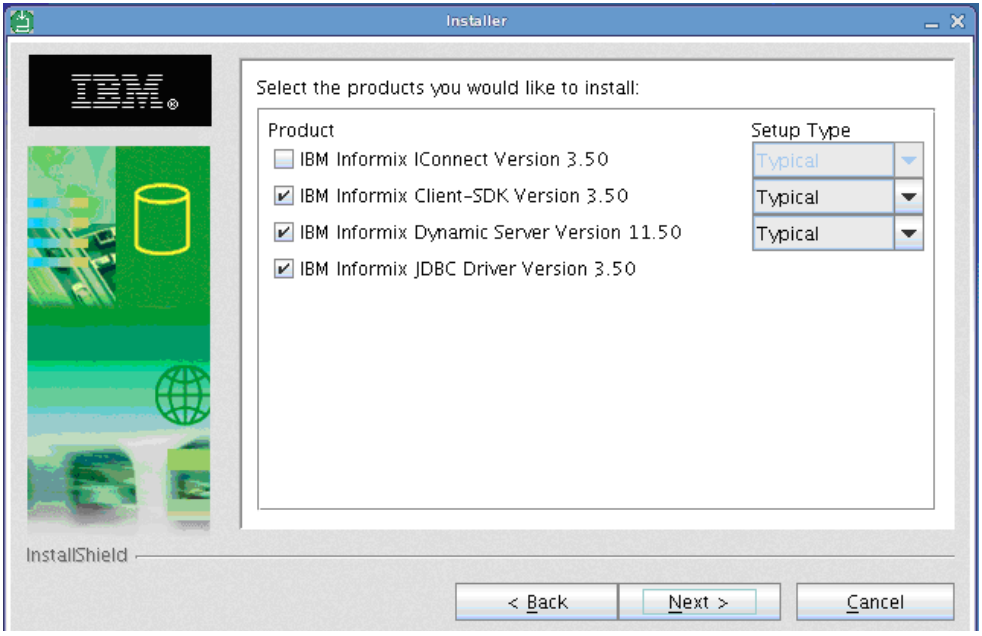


Figure 3: IDS on Linux Product Selection

At this point the installer prepares to install the database server. After a welcome screen to the IDS 11.50 installation, you will have to accept a license agreement before the installation takes place.

The next few screens identify the type of installation to use and the directory where to install the software. Please make sure to select the “typical” installation and keep the default installation directory if possible.

The default directories are:

- On Windows: C:\Program Files\IBM\IBM Informix Dynamic Server\11.50
- On Linux: /opt/IBM/Informix
- On MAC OS X: /Applications/IBM/Informix

On Windows, if the Informix user does not exist, you will be asked to provide a password for it.

On the screen asking for role separation, make sure to select “No”.

On UNIX-type systems, you will be asked if you want to create a demonstration database server as shown in figure 4. Make sure you select “Yes”.

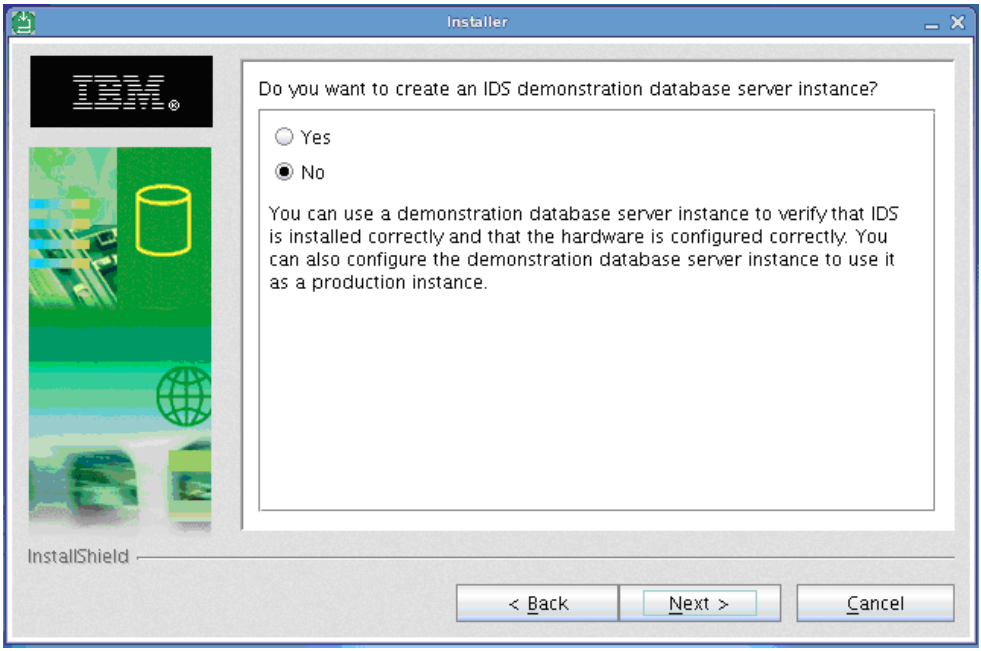


Figure 4: Demonstration Instance Creation

The following screen asks you which configuration file to use. Select “Use the default configuration file”.

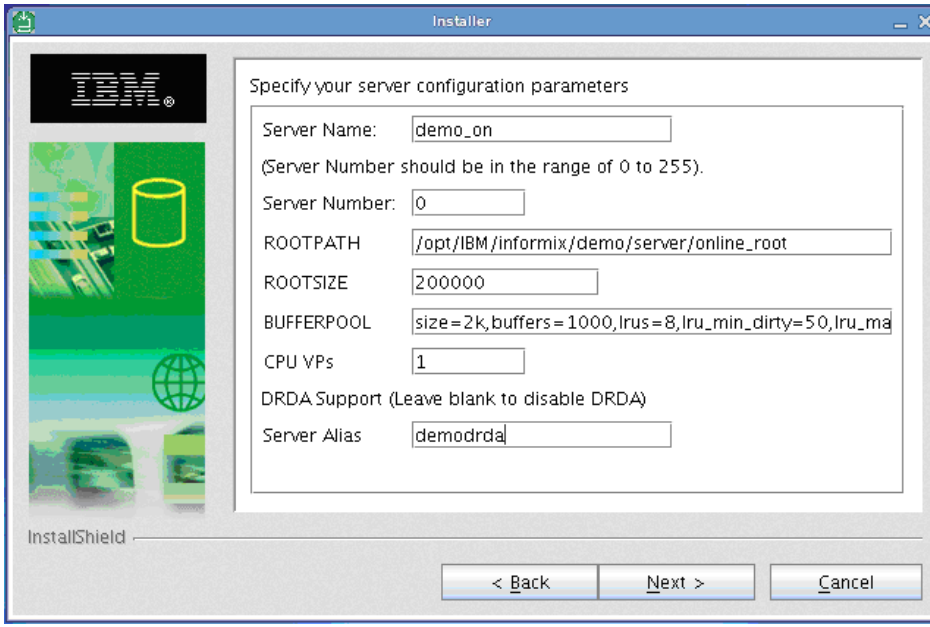


Figure 5: Configuration Parameters

On the UNIX-type systems, you should see a screen similar to Figure 5. You can keep the defaults provided. Make sure to add `demodrda` in the Server Alias field. You will go through five more screens. Just stay with the default values.

On Windows, you will have to answer other screens for the installation of the other products. Just take the default values. If you are using Microsoft Visual Studio, you may want to launch the VSAI Installer when reaching that screen.

Server Setup

At this point a Windows installation uses the following server configuration:

Server name: ol_ids_1150_1

Service port name: svc_ids_1150_1

Service port number: 9088

Server number: 0

Server alias: svc_drda

Alias port: 9089

On UNIX-type systems, the values are as follows:

Server name: demo_on

Service port name: sqlexec

Service port number: 9088

Server number: 0

Server alias: demodrda

Alias port: 9089

These values are used for client applications to connect to IDS.

Setting Up the Environment

Now that the server is running, you can access it by setting the environment properly. The Windows environment gives you this setup automatically by using the Start menu:

Start→All Programs →IBM Informix Dynamic Server 11.50→ol_ids_1150_1

On UNIX-type systems, you can use a script to setup the environment. Assuming that the installation directory is $INFORMIXDIR, you can execute:

- For bourne shell:

```
. $INFORMIXDIR/demo/server/profile_settings
```

- For C shell:

```
source $INFORMIXDIR/demo/server/profile_settings.csh
```

These scripts set the following environment variables:

- `INFORMIXSERVER`: Name of the server as listed in the server setup section above
- `INFORMIXDIR`: Informix installation directory
- `ONCONFIG`: Name of the configuration file
- `INFORMIXSQLHOSTS`: Name of the file containing server connectivity information
- `PATH`: Adds the `$INFORMIXDIR/bin` directory to have access to the Informix executables

You should include these values in the login script that sets up your environment. This way, your environment will always be set properly to use IDS.

Bringing Up the Database Server

After the installation, the IDS server will be up and running. On Windows, it is set up as a service so the server will be brought up automatically when you start your system. You can edit the service (Informix IDS – ol_ids_1150_1) properties if you don't want it started automatically.

On UNIX-type systems, you can add to the boot sequence by including the appropriate script to the desired run level. For more information, see the UNIX/Linux documentation for `init.d`.

You can start IDS as root or as Informix by executing:

```
$ oninit
```

The database server will be available for use in a few seconds. You can monitor the status of the server by issuing:

```
$ onstat –

IBM Informix Dynamic Server Version 11.50.UC2DE, -- On-Line
– Up 00 :13 :09 – 38208 Kbytes
```

Once you see the status "On-Line" the server is ready for use.

So, the "start" section of the init script would issue the oninit command. You should have the start section

execute at run levels three and higher (likely 3 and 5). You should then have the following scripts:

```
/etc/init.d/rc3.d/S14informix
/etc/init.d/rc5.d/S14informix
```

Shutting Down the Database Server

In the Windows environment, the database server will shut down when you shut down your machine. You can also shut down the database server by selecting the Informix service and stopping it.

On UNIX-like systems, you can shut down the server with the following command executed as root or Informix:

$ onmode –ky

To automate the shut down of the server as you shutdown the system, you need to add the appropriate stop functions in the run levels 3 and 5:

```
/etc/init.d/rc3.d/K08informix
/etc/init.d/rc5.d/K08informix
```

Automate Run Level Script

The script used to bring up or down IDS as your system comes up or is shut down would be as follows:

```
#!/bin/sh
INFORMIXSERVER=demo_on
INFORMIXDIR="/opt/IBM/informix"
ONCONFIG=onconfig.demo_on
INFORMIXSQLHOSTS="/opt/IBM/informix/etc/sqlhosts"
PATH=${INFORMIXDIR}/bin:${PATH}
export INFORMIXSERVER INFORMIXDIR ONCONFIG INFORMIXSQLHOSTS
PATH
if [ $# -lt 1 ]
then
  echo "Usage: $0 {start|stop}"
else
  case "$1" in
    'start')
        if [ `$INFORMIXDIR/bin/onstat 2>&- | grep -c On-Line` -ne 1 ]
        then
          rm -f /INFORMIXTMP/*
          rm -f /opt/IBM/informix/demo/server/online*.log
          rm -f /opt/IBM/informix/tmp/*
          echo -n "Starting Informix Dynamic Server V11.5
Developer Edition..."
          $INFORMIXDIR/bin/oninit
          echo "done"
        fi
    ;;
    'stop')
        if [ `$INFORMIXDIR/bin/onstat 2>&- | grep -c On-Line`
-eq 1 ]
        then
          echo -n "Shutting down Informix Dynamic Server V11.5
Developer Edition..."
          $INFORMIXDIR/bin/onmode -ky
          echo "done"
        fi
    ;;
    *)
        echo "Usage: $0 {start|stop}"
    ;;
```

```
  esac
fi
```

Adding Space to the Database Server

At this point, you have a running database server with some space allocated to it. Depending on the amount of space you need, you may have to add to the allocation.

Another reason to add space to the server is if you are planning to use large object (types BLOB and CLOB[1]).

IDS uses logical spaces called dbspaces to store database information. IDS also has specialized dbspaces called sbspace to store BLOBs and CLOBs. Other specialized dbspaces exists for temporary space, blob spaces[2] and external spaces. We will limit our discussion to dbspaces and sbspaces.

A dbspace includes one or more chunks. A chunk is a piece of physical storage such as a file or a disk device. You can either create additional spaces or add storage to existing ones.

After the installation the space allocation is as follows:

Platform	Dbspace name	size	Free space
Windows	`rootdbs` `ol_ids_1150_1`	200MB 100MB	75MB 99MB
UNIX-type	`rootdbs`	200MB	76MB

You can find the files allocated to the dbspaces as follows:

Platform	Path
Windows	C:\IFXDATA\ol_ids_1150_1\rootdbs_dat.000 C:\IFXDATA\ol_ids_1150_1\ol_ids_1150_1_dat.000
Linux	/opt/IBM/Informix/demo/server/online_root
MAC OS X	/Application/IBM/Informix/online_root

It is good practice to leave the `rootdbs` dbspace for system objects and use a separate dbspace for user objects, including additional databases.

Adding a dbspace

The Windows environment already has an additional dbspace so we don't need to add anything. The creation of a dbspace is done in three steps:

1.) Create the file that will be used:
```
touch $INFORMIXDIR/demo/server/dbspace1
```

2.) Set the proper owner, group, and permission on the file:
```
chgrp informix $INFORMIXDIR/demo/server/dbspace1
chown informix $INFORMIXDIR/demo/server/dbspace1
chmod 660 $INFORMIXDIR/demo/server/dbspace1
```

3.) Allocate the file to IDS:
```
onspaces -c -d dbspace1 -p $INFORMIXDIR/demo/server/dbspace1
-o 0 -s 102400
```

1. BLOB: Binary Large Object – CLOB: Character Large OBject
2. blob spaces are used for the older types of large objects, SQL types TEXT and BYTE

The `onspaces` command gives a name to the dbspace `(dbspace1)`, identifies the device/file that is used `($INFORMIXDIR/demo/server/dbspace1)`, the offset (0), and the size in kilobytes (100MB).

Adding an sbspace

If you want to use the BLOB and CLOB types, you need to create an sbspace. It is done the same way as creating a dbspace with a slightly different syntax. Note that the example shows the execution on a UNIX-like platform. On Windows, replace the path used with
`"C:\IFXDATA\ol_ids_1150_1\sbspace1.dat"`:

4.) Create the file that will be used:
```
touch $INFORMIXDIR/demo/server/sbspace1
```

5.) Set the proper owner, group, and permission on the file:
```
chgrp informix $INFORMIXDIR/demo/server/sbspace1
chown informix $INFORMIXDIR/demo/server/sbspace1
chmod 660 $INFORMIXDIR/demo/server/sbspace1
```

6.) Allocate the file to IDS:
```
onspaces -c -S sbspace1 -p $INFORMIXDIR/demo/server/sbspace1
-o 0 -s 102400
```

As you can see, the only difference in the syntax is the use of "-S" instead of "-d".

When using BLOBs and CLOBs, you usually need to identify where they should be stored in the CREATE TABLE statement. You can identify a default location for their storage in the IDS configuration file. It turns out that the configuration file already identifies the subspace called "`sbspace1`" as the default storage. Since we've used this name in the `onspaces` command above, we don't need to do anything more. For your information, the configuration file can be found at the following location:

- Windows: `%INFORMIXDIR%\etc\%ONCONFIG%`
- Other: `$INFORMIXDIR/etc/$ONCONFIG`

The parameters are: `SBSPACENAME` and `SYSSBSPACENAME`. Set them as follows:

```
SBSPACENAME    sbspace1 # Default sbspace
SYSSBSPACENAME sbspace1 # Default System sbspace
```

Creating a Database in the Server

We are now ready to use the database server. The first step is to create a database. Since we'll need to test connectivity in any programming environment we use, we need a know database for that testing. IDS has a utility that allows you to create such a database. In a command window with the environment setup properly as discussed in the "Setting Up the Environment" section above, the following command creates the stores database in `dbspace1`:

```
$ dbaccessdemo7 stores -log -dbspace dbspace1
```

You may also want to create your own database for your development project. You could do this through an application, but it is much simpler to use the `dbaccess` utility. This utility works in screen mode and also in command line mode. You can also submit a script to `dbaccess` for execution. You can find more information on `dbaccess` in the dbaccess manual listed in the reference section at the end of this chapter.

To create a database called `mydb`, you can use the following command:

```
CREATE DATABASE mydb in dbspace1 WITH BUFFERED LOG;
```

The database `mydb` is created in the `dbspace1` dbspace. This means that, by default, the tables will also be created in that dbspace. The `BUFFERED LOG` is to insure logging but not as strictly as without the `BUFFERED` keyword. Since we are setting up a development system, crash recovery does not have to be as good. Assuming that the command is in a file called `cre8.sql`, you can execute it by issuing the following command:

```
$ dbaccess –e – cre8
```

The "-e" option tells dbaccess to echo the commands to the screen (standard output). The following "-" simply says not to open any database but only connect to the server. Note also that the ".sql" extension is assumes so it is not necessary to add it to the command file name argument. You can use the dbaccess command to submit ant type of SQL statements to a database. For example, assuming that you have a set of SQL statements to create tables in a file called `cre8tabs.sql`, you can create the tables in the database mydb with the following command:

```
$ dbaccess –e mydb cre8tabs
```

At this point, we are ready to move on to setting up the environment for a specific programming language for application development.

Reference Material

- *IBM Informix Dynamic Server Installation Guide for UNIX, Linux, and MAC OS X, Version 11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=GC23-7752-01
- *IBM Informix Dynamic Server Installation Guide for Windows, Version 11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=GC23-7753-01#
- *IBM Informix Dynamic Server Administrator's Guide, v11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-7748-01
- *IBM Informix Dynamic Server Administrator's Reference, v11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-7749-01
- *IBM Informix Dynamic Server DB-Access User's Guide, v11.5*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-9430-00
- *Administering Informix Dynamic Server,* Carlton Doe, Sept 2008,
  ISBN 978-1-58347-076-3

3

Installing a Stand-Alone Client

There are two separate client software development kits (Client-SDK) available for IBM Informix Dynamic Server (IDS). The Informix Client SDK is specific to Informix and the other, the IBM Data Server Client, is a package that is designed to work with IBM's database servers, including DB2® on all platforms and IDS. The long term direction is with the IBM Data Server Client.

If you are planning to do software development with the Informix Client SDK on the same machine as the one where you installed IDS, the Informix Client SDK is already installed and you can skip this chapter. If you are planning to use the IBM Data Server Client, you need to install it.

This chapter is divided in three parts. The first part discusses choosing the client SDK that will answer your needs, the second part discusses how to obtain and install the IBM Data Server Client and a third part does the same for the Informix Client SDK.

Choosing a Software Development Kit

The first issue that impacts your choice is platform support. For example, the IBM Data Server Client is not available on the MAC OS platform. If the platform is not an issue, the choice of which client SDK to use comes down mainly to the functionality supported. The IBM Data Server Client does not support the following types:

- Complex types: ROW, SET, MULTISET, LIST
- Opaque types
- Interval type
- Boolean is seen as smallint

Despite this limitation, it may be completely adequate to use this SDK for your environment. You can even use these types in your schema as long as you don't use them through the SDK. For example, this can be accomplished through the use of casting. If you really need to use these restricted types such as in spatial type applications, your current choice would then be the Informix client SDK.

The IBM Data Server Client supports all the drivers discussed in this book. The Informix Client SDK does not support the open-source Ruby and Rails drivers. This does not mean that the drivers are not working. It simply is that IBM does not provide support for them. If the application you are developing will eventually go into production, the IBM Data Server Client is likely a better choice.

If you are still not sure about which one to use, you should then install the IBM Data Server Client and start with this interface. You can always switch to the Informix Client SDK if needed. The application changes should be minimal in most environments.

IBM Data Server Client

The IBM Data Server Client comes in multiple flavors:

- IBM Data Server Driver for JDBC and SQLJ.
  This packaging offers support for Java application requiring the use of JDBC or SQLJ. If you are

planning to only do Java development, please refer to Chapter 4 for more information in downloading and installing this driver.

- IBM Data Server Driver for ODBC and CLI.
  This is a driver that provides runtime support for ODBC and CLI applications. It also supports the XA API. Messages are reported only in English.
- IBM Data Server Driver for ODBC, CLI and Open Source and IBM Data Server Driver for ODBC, CLI, and .NET
  These drivers provide runtime support for applications using ODBC, CLI, .NET, JDBC and SQLJ, and open source drivers (PHP, Ruby) for lightweight deployment.
- IBM Data Server Runtime Client
  This packaging includes everything mentioned in the previous bullet plus some additional utilities
- IBM Data Server Client
  This packaging includes everything that is in the IBM Data Server Runtime Client plus several utilities and application header files. It also includes support for traditional language development such as C and FORTRAN.

To insure that you have everything you need for your development environment, you should download the IBM Data Server Client. The current release at the time of this writing is Version 9.5. You can find the download link at:

```
https://www14.software.ibm.com/webapp/iwm/web/reg/pick.do?l
ang=en_US&source=swg-idsc11
```

You will need to register and log in to have access to the download. The download size is:

- Windows: 287MB .zip file
- Linux: 515MB .tar.gz file

If you are planning to only use the Java interface, you don't need to install the entire IBM Data Server Client. Go directly to Chapter 4.

For the rest of this book, we assume that you install the IBM Data Server Client since there may be a need to recompile the open-source drivers or use the Visual Studio add-in.

If you are planning to do development in PHP or Ruby/Rails, you may be able to use a smaller driver package: IBM Data Server Driver for ODBC, CLI, and Open Source. You can find this package on the IBM support site at:

```
http://www-01.ibm.com/support/docview.wss?rs=71&uid=swg21288110
```

You simply have to pick the appropriate platform and select the driver. For UNIX-type platforms, it comes as a compressed .tar file (.tar.gz). On Windows, it comes as an executable (.EXE).

For .NET development, you may be able to use the same web site and download the IBM Data Server Driver for ODBC, CLI, .NET and Open Source. You will also need to download the IBM Database Add-In for Visual Studio. Look at the following site for the download:

```
http://www-01.ibm.com/software/data/db2/ad/dotnet.html
```

Data Server Client Installation

The first step is to extract the content of the download files:

- On Windows: unzip the file
- On Linux: tar xzvf <filename>

The extraction creates a directory called client. Move to that directory and execute:

- On Windows: `image/setup.exe`
- On Linux: `./db2setup` for GUI installation or `./db2_install` for command line installation.

At this point, for the GUI installation, you get a welcome screen similar to the one shown in figure 1. From this point, the installation is pretty straightforward so there is no reason to go through all the details of each screen. There is one place where it may be a bit confusing:

- When you see the screen “Select the Installation Type”, make sure to select “Typical”.

The default installation directory is:

- On Windows: `C:\Program Files\IBM\SQLLIB`
- On Linux: `$HOME/sqllib` where `$HOME` is the home directory of the account doing the installation.

In the Windows environment, you also have the option of installing the IBM Database Add-Ins for Visual Studio 2005. If you are using Visual Studio, you should install the add-in.

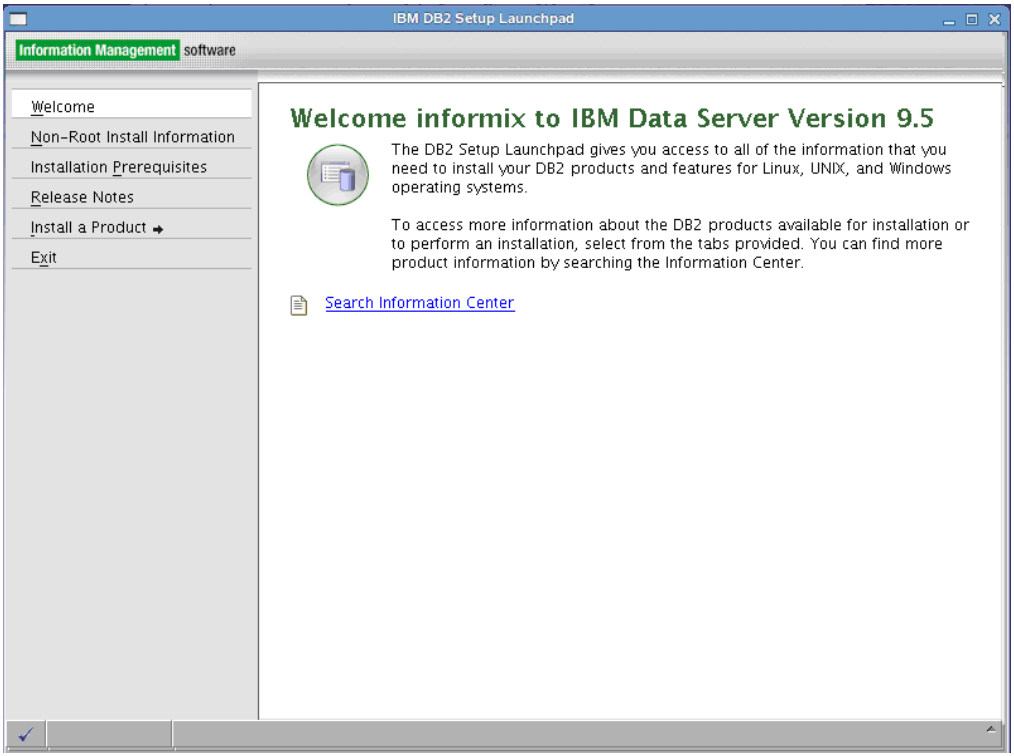


Figure 1: Installation Welcome Screen

Setting up the Environment for Development

In the Windows environment, the installation made the necessary modifications to the environment so you can use the IBM Data Server Client. On UNIX-type systems, you need to set up the environment:

- Bourne shell: `.$HOME/sqllib/db2profile`
- C Shell: source `$HOME/sqllib/db2cshrc`

Since you need this setup each time you use the IBM Data Server Client, you may want to add this to your login procedure (`ex: /etc/profile`).

At this point, you are ready to move on to setting up your environment for the language of your choice.

Informix Client SDK

The Informix client SDK includes the following products:

- IBM Informix Object Interface for C++: A C++ interface to develop object-oriented client applications for use with all IDS and client-side value objects for IDS
- IBM Informix ESQL/C: An SQL embedded-language product that is used to create custom C applications
- BM Informix Global Language Support (GLS): An interface that allows IBM Informix products to use different locales that have defined conventions for a particular language, culture, or code set
- IBM Informix ODBC Driver: An Informix implementation of the Open Database Connectivity (ODBC) 3.0 Level 1+ standard that supports Microsoft(R) Transaction Server (MTS)
- IBM Informix .NET Provider (Windows only): A .NET assembly that lets .NET applications access and manipulate data in IDS databases. Requires .NET Framework, Version 2.0 or 1.1
- IBM Database Add-Ins for Visual Studio 2005 (Windows only): A collection of add-ins to the Microsoft Visual Studio .NET IDE. The add-ins simplify the creation of applications that use the ADO.NET interface. Requires Microsoft Visual Studio 2005
- IBM Informix OLE DB Provider: A client-side, native OLE DB provider that implements full functionality for base-level providers and contains extensibility support for IDS
- IBM Informix Connection Manager: Performs connection routing and failover for high-availability clusters
- LIBMI Client API: The DataBlade API for client applications
- Simple Password Communication Support Module (SPWDCSM): Provides password encryption for client applications
- Generic Security Services Communication Support Module (GSSCSM): Provides single sign-on authentication for client applications
- IBM Informix Setnet32 utility (Windows only): A utility for setting environment variables and configuring connections to the database server
- DSN Migration utility (Windows only): A utility for migrating or restoring an Intersolv ODBC DSN to an Informix ODBC DSN
- GSKit: Utilities to create and manage keystores and digital certificates for Secure Sockets Layer protocol encryption
- Ilogin Demo (Windows only): A utility to test connections to IDS
- finderr (UNIX) and Find Error (Windows): A utility to find information about IBM Informix-specific error messages, including corrective actions
- Documentation Viewer (Windows only): A utility to view client release and machine notes in text format

You need the client SDK to compile the drivers for languages such as PHP and Ruby. The current release at the time of this writing is version 3.50.xC3.

The JDBC driver is in a separate download. This is covered in Chapter 4.

Getting Informix Client SDK

The Informix Client SDK is available for download at:

```
http://www14.software.ibm.com/webapp/download/search.jsp?rs
=ifxdl
```

The Informix Client SDK is a zip file for Windows and a tar file for Linux and MAC OS X. Make sure to select the appropriate version for the platform you are using. You will need to register and sign in to complete the free download. The download is around 70MB in size.

Informix Client SDK Installation

After you download the archive file you need to extract the content to a temporary directory:

- On Windows: unzip the file
- Linux and MAC OS: `tar xvf <filename>`

The installation is done by executing the installation script:

- On Windows: `setup.exe`
- On Linux and MAC OS: `./installclientsdk -gui`

You must be root to install Informix Client SDK on UNIX-type systems.

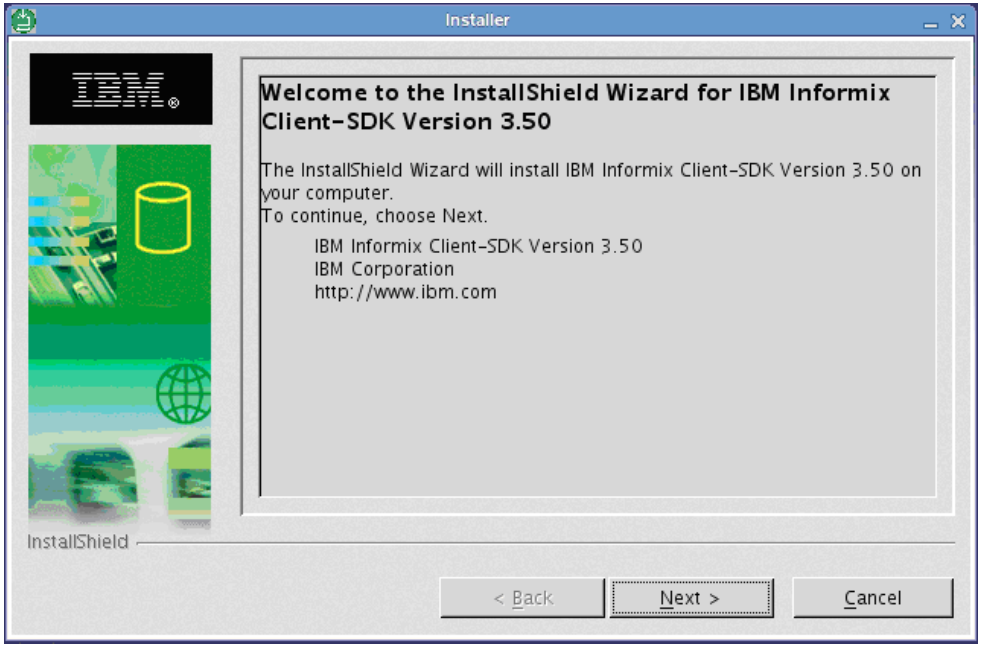


Figure 2: Informix Client SDK Installation Welcome Screen

Figure 2 shows the welcome screen of the Informix Client SDK installation in graphic mode.

The installation itself is self-explanatory. The only part you need to remember is the installation directory:

- On Windows: `C:\Program Files\IBM\Informix\Client-SDK`
- On Linux: `/opt/ibm/Informix`
- On MAC OS: `/Applications/IBM/informix`

Setting Up the Environment for Development

You need to make sure you have references to the different components of the Informix client SDK. This should include:

- Setting the `INFORMIXDIR` environment variable to the installation directory
- Setup the IDS server information
- Adding the `$INFORMIXDIR/bin` to the `PATH` environment variable

On Windows, you should only have to set the IDS server information. This is done with the setnet32.exe program. On Unix-type platforms, you may want to set the environment variables so they are set when you log in. You also need to create the `sqlhosts` file to identify the IDS server information. The next two sections cover the set up.

Setting up on Windows

The Setnet32 program can be executed from the start menu. This is shown in figure 3.

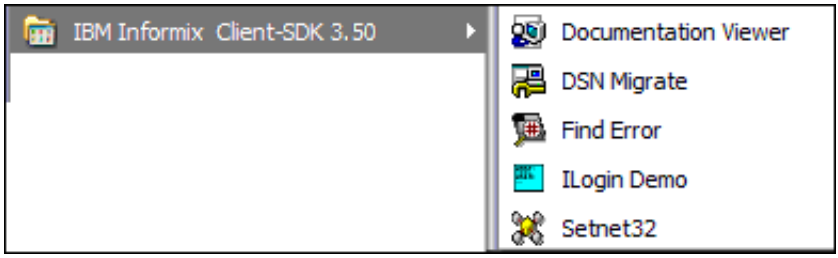


Figure 3: Selecting the Setnet32 program

The program execution brings up a window as shown in figure 4. Under the Server Information tab, you provide information about the IDS database server. This includes:

- **IBM Informix Server:** This is the name of the IDS server. If you followed the IDS installation of Chapter 2, the name is `demo_on`.
- **HostName:** This is either the host name or host address of the machine where the IDS server is running. If you use a hostname, you have to make sure that name is known on your machine so the address can be resolved.
- **Protocolname:** this is the internet protocol used. Use the default `onsoctcp`.
- **Service Name:** this is either a service name or number. If you use a service name, you must have this name defined on your machine in the file `C:\WINDOWS\system32\etc\services`.

These values are saved in the Windows registry. You can use the `ilogin` demo program to test the connectivity with the IDS server.

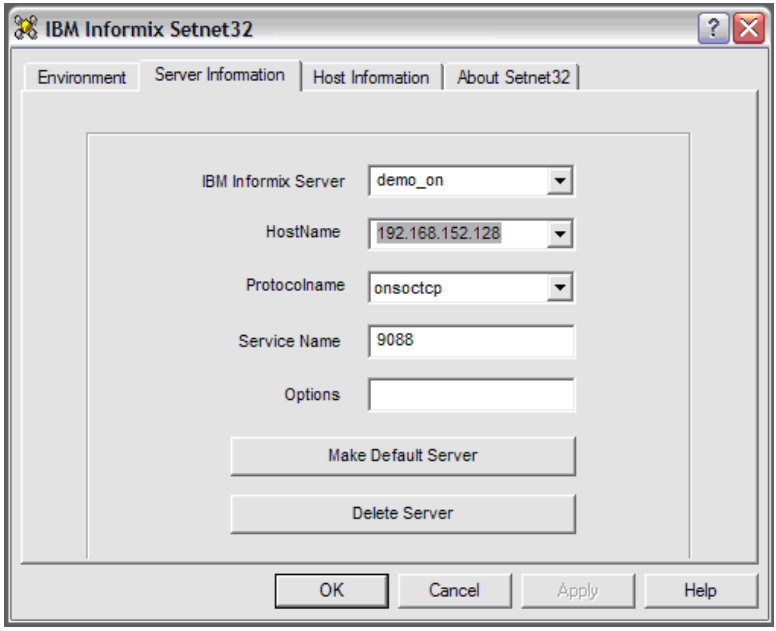
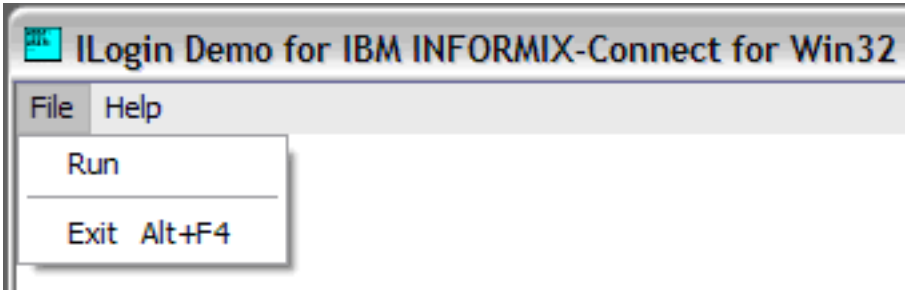
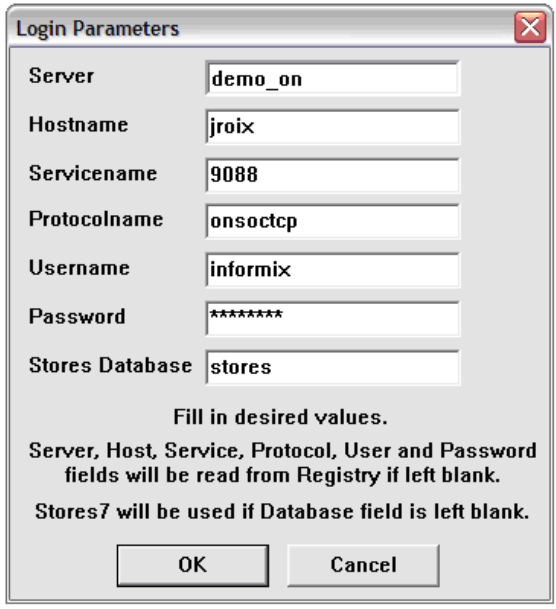


Figure 4: Setnet32 Program Execution

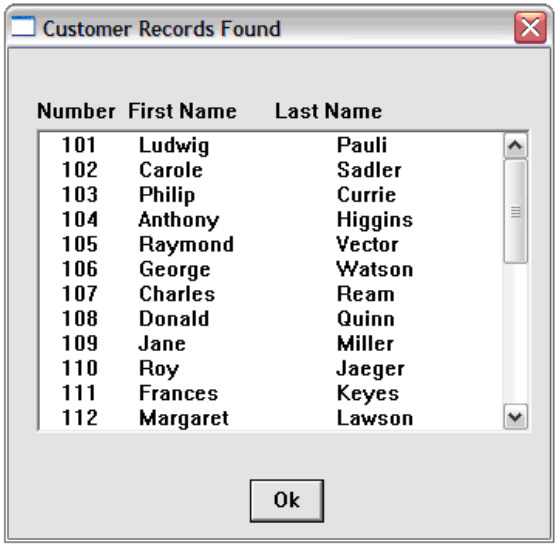
The `ilogin` demo program executes a select statement on a specific table. You should use the stores demo database that was created as part of the installation in Chapter 2. After starting the `ilogin` program, select File → Run:



What comes next is a login parameter window:



Once the parameters are filled out, press OK. If the connection is successful, it should display the content of the customer table:



You can also test the database connectivity by writing your own test program using the programming environment of your choice. This is basically what is demonstrated in Chapters four to seven.

Setting Up on Unix-Type Machines

The sqlhosts file is assumed to be located in the `$INFORMIXDIR/etc` directory. This file includes the following information:

- **IDS server name:** this is the name that you use to connect to. It ties all the other pieces of information together.
- **Protocol:** should be `onsoctcp`.
- **Hostname:** This is the hostname or host address. If you use a hostname, you have to make sure it is known on your machine so the name can be resolved to an address.
- **Service:** either a service name or a port number. This value corresponds to the port the IDS server is listening to. If you use a name, it must be defined in `/etc/services`.

Assuming that the IDS server is on a machine named `jroix`, based on the server installation done in Chapter 2, the `sqlhosts` file should contain the following:

```
demo_on         onsoctcp  jroix        9088
```

You are now ready to set up your programming environment.

Reference Material

- *Informix Client Software Development Kit and Informix Connect Web Page*
  http://www-01.ibm.com/software/data/informix/tools/csdk
- *IBM Informix Client Products Installation Guide, v3.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=GC23-9413-01#
- *Quick Beginnings for IBM Data Server Clients, DB2 version 9.5*
  ftp://ftp.software.ibm.com/ps/products/db2/info/vr95/pdf/en_US/db2ite951.pdf
- *Getting Started with Database Application Development, DB2 version 9.5*
  ftp://ftp.software.ibm.com/ps/products/db2/info/vr95/pdf/en_US/db2axe951.pdf
- *IBM DB2 9.5 Information Center*
  http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp?topic=/com.ibm.db2.luw.qb.client.doc/doc/c0022612.html

4

The Java Environment

In this chapter we look at how we can use Informix Dynamic Server (IDS) with the Java programming language. We cover the basics of which drivers are available, where to get the drivers and how to install and use. Information on how to connect to IDS using these drivers follows.

There is additional information provided on data type mapping, use of IDS with an application server and available development tools.

JDBC and IDS

Since IDS supports two communication protocols (SQLI and DRDA), there are two different JDBC drivers available:

- IBM Data Server driver for JDBC and SQL/J: This driver supports both DB2 and IDS. It is a type-4 driver that is compliant with the JDBC 3.0 and JDBC 4.0 functions. The latest driver is version 3.52. It supports IDS starting with version 11. In this chapter, we refer to this driver as the Data Server driver.
- Informix JDBC 3.5: This type-4 JDBC driver is compliant with the JDBC 3.0 specifications. It is also extended to support the extensibility types included in IDS as well as any user-defined types that are created. In this chapter we refer to this driver as the Informix driver.

The discussion on software development kits in Chapter 3 applies to the JDBC drivers. There are some data types that are not supported by the Data Server driver but you may not have any plans to use those anyway. The long term direction for Informix drivers is to use the common drivers such as the Data Server driver. Refer to the mapping section below to for details on supported types.

Obtaining and Installing the Data Server Driver

The data server driver does not come with IDS so you need to download it from the IBM site. The Data Server driver is included in the DB2 9 client software but it can also be downloaded separately. You can find a link for the download at the bottom of the page at the following URL:

```
http://www-01.ibm.com/software/data/db2/ad/java.html
```

The link is titled “IBM Data Server Driver for JDBC and SQLJ”. The driver comes as a 6MB zip file. The installation simply consists of extracting the content from the zip file. On Linux, you can use the following command:

```
$ unzip db2_db2driver_for_jdbc_sqlj.zip
```

With the latest driver, the extraction creates a directory named `db2_db2driver_for_jdbc_sqlj`. You can move the content of this directory to a location of your choice if you so desire.

Obtaining and Installing the Informix Driver

First, note that the latest Informix JDBC driver requires the Java JDK 1.4.2 or later. This should not be a problem since the current JDK release is version 1.6.

If you have installed IDS on your machine as described in Chapter 2, the Informix JDBC was installed as part of the client software development kit (C-SDK) installation. The driver can be found in the Informix

installation directory, under the `jdbc` directory. If this is the case, you can skip the rest of this section.

The Informix JDBC driver can be downloaded for free from the IBM site. The easier way to get it is to go the Informix JDBC driver page at:

```
http://www-01.ibm.com/software/data/informix/tools/jdbc/
```

You can find a link in the bottom right section of the page in a box titles “Trials and Demos”. The download is free but you will need to register to get to it. At the time of this writing, the current Informix JDBC driver is JDBC 3.50.JC2. It is a 17MB .tar file download.

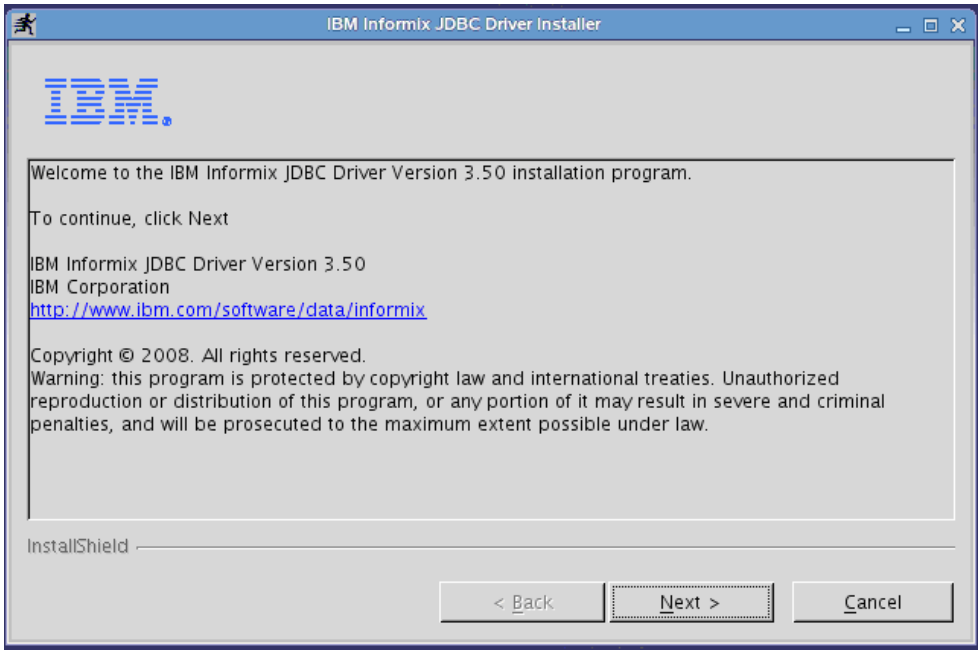
The first step is to extract the driver from the .tar file. On Unix-type platforms, you can issue the following command:

```
$ tar xvf JDBC.3.50.JC2DE.tar
```

On the Windows platform, you can use a utility such as “winrar” to extract the content. The next step is to execute the setup.jar file to start the installation. Use the following command:

```
$ java -cp setup.jar run
```

You will be guided for the installation through screens starting with the one shown here:



The installation consists of accepting the license agreement and providing the installation directory.

To use the installation in terminal mode, add the “`-console`” option on the command. For more information on the installation options such as silent install, please refer to the Informix JDBC driver programmer’s manual listed at the end of this chapter.

The only decision you have to make during the install is where the software is installed. The installation

software provides a default value that you can change.

Trying the Drivers

Before we continue, let's make sure that we can now connect to IDS. The first thing to do is to add the drivers to the `CLASSPATH` variable. Let's assume that the Data Server driver was installed/unzipped in the `$JDBC_HOME` directory and the Informix driver is under `$INFORMIXDIR/jdbc` directory. The `CLASSPATH` variable should contain:

```
$ CLASSPATH=$JDBC_HOME/db2_db2driver_for_jdbc_sqlj/
db2jcc.jar:$INFORMIXDIR/jdbc/lib/ifxjdbc.jar:.
```

In the Windows environment, you can set the CLASSPATH using the following statement:

```
> set CLASSPATH=%JDBC_HOME%\db2_db2driver_for_jdbc_sqlj\
db2jcc.jar:%INFORMIXDIR%\jdbc\lib\ifxjdbc.jar;.
```

We can test the connectivity of both drivers using the following `Connect2IDS.java` class:

```
import java.util.*;
import java.sql.*;

public class Connect2IDS {
  private static String URL[] = {
  "jdbc:ids://jroix:9089/stores",
  "jdbc:informix-
    sqli://jroix:9088/stores:INFORMIXSERVER=demo_on"
  } ;
public static void main( String[] args)
       throws SQLException, ClassNotFoundException {

    Connection conn;
    ResultSet rs;

    Class.forName("com.ibm.db2.jcc.DB2Driver");
    Class.forName("com.informix.jdbc.IfxDriver");

    for (int i = 0; i < 2; i++) {
      System.out.println("Testing URL: " + URL[i]);
      conn = DriverManager.getConnection(URL[i],
                           "informix", "informix");
      if (conn != null) {
        System.out.println("Got a connection to the DB");
        String [] tableTypes = { "TABLE", "VIEW" };
        DatabaseMetaData dbmd = conn.getMetaData();
        rs = dbmd.getTables(null, null, null, tableTypes);
        while (true == rs.next()) {
          System.out.println(
               rs.getString(2) + "." + rs.getString(3) +
               " [" + rs.getString(4) + "]");
     }
        conn.close();
      } else {
        System.out.println(
               "Did not get a connection to the DB");
      }
    }
    return;
  }
}
```

You will need to replace the hostname `(jroix)` with the name of the host where IDS was installed.

This class first loads both drivers then tries to connect to the IDS database using each driver in turn. The first driver used is IBM Data Server driver. It is selected through the connection URL:

```
jdbc:ids://jroix:9089/stores
```

The format of the URL is as follows:

```
jdbc:ids://{ip-address or host-name}:{port-number}/{database name}:
[name=value[;name=value]. . .]
```

The beginning of the URL, "`jdbc:ids`", identifies the driver to use: IBM Data Server. The following sections represent the host, port number and database. Note that in our example, we stick to the minimum information needed. We do not provide properties (name=value).

The process is repeated with the Informix JDBC driver. It is selected through the connection URL:

```
jdbc:informix-
sqli://jroix:9088/stores:INFORMIXSERVER=demo_on
```

The format of the URL is:

```
jdbc:informix-sqli://[{ip-address|host-name}:{port-
number|service-name}][/dbname]:
INFORMIXSERVER=servername[{;user=user;password=password]
|CSM=(SSO=database_server@realm,ENC=true)}
[;name=value[;name=value]...]
```

As you can see we have used the minimum information to connect to IDS. For more information on the format of database URLs, please consult the IBM Informix JDBC Programmer's Manual listed in the reference section.

Once a connection is established, the program displays the database tables and views.

The program is compiled and executed using the following statements:

```
$ javac Connect2IDS.java
$ java Connect2IDS
```

The program should work without any problems. A message indicating that a class was not found means that your `CLASSPATH` variable is not set properly. If the program does not connect to IDS, it is either related to the URLs used that do not point to the proper database server or database, or it is because of improper username/password used in the `getConnection` method.

At this point, you should have been able to connect to IDS and you are ready to move on.

More on the IBM Data Server Driver

The IBM Data Server driver includes three files:

- `db2jcc.jar`: JDBC driver that follows the JDBC 3.0 specifications
- `db2jcc4.jar`: JDBC driver that follows the JDBC 3.0 specifications with additional JDBC 4.0 functions
- `sqlj.zip`: Zip file used for SQLJ programs. Put it as is in the `CLASSPATH`

This driver includes functions to monitor your application performance. The data available includes core

driver time, network I/O time, server time and application time. Please consult the documentation for more information.

More on the Informix JDBC Driver

When you install the Informix JDBC driver, you end up with the following directories:

- `demo`: Sample programs that use the JDBC API. It is described in more details in the IBM Informix JDBC Driver Programmer's Manual listed in the reference section
- `doc`: IBM Informix JDBC Driver Programmer's manual in PDF format, SQLJ HTML documentation and Javadoc documentation
- `lib`: JDBC interface jar files
- `license`: Text of the license agreement in multiple languages
- `proxy`: Includes three classes to support http tunneling proxy class
- `_uninst`: Contains a jar and a dat file used to uninstall the JDBC driver

The lib directory contains multiple jar files that are needed for more complex situations. In many cases, only ifxjdbc.jar will be needed. The additional files are: `ifxjdbcx.jar`, `ifxlang.jar`, `ifxlsup.jar`, `ifxsqlj.jar`, `ifxtools.jar`. Please see the documentation for more information.

There are many things you can do to optimize your application communication with the database server. It includes areas such as communication buffer sizes, use of bulk insert, minimize the granularity of communication, use of an LDAP directory, and so on. Please consult the information to understand all you can do with the Informix JDBC driver.

Using a DataSource to Connect to IDS

A DataSource is usually used to establish a connection, potentially giving the benefit of being more portable and allow for the pooling of database connections for reuse. It is the connection method of choice in application servers.

The DataSource interface is usually used in the context of an application server. We cover this later in this chapter.

Type Mapping

To help you get started with manipulating database objects, here is a type mapping table between Java and the two available drivers. There are sometimes multiple types that can be used for one SQL type. Only one is listed in the table. Please consult the documentation for more information.

Database Type	Informix JDBC driver Java types	IBM Data Server driver Java type
BIGINT, BIGSERIAL	long	?? Long ??
BLOB	Java.sql.Blob	java.sql.Blob
BOOLEAN	boolean	boolean
BYTE	java.io.InputStream	byte[ ]
CHAR	java.lang.String	java.lang.String
CLOB	java.sql.Clob	java.sql.Clob

DATE	java.sql.Date	java.sql.Date
DATETIME HOUR TO SECOND	java.sql.Time	java.sql.Time
DATETIME YEAR TO FRACTION(5)	java.sql.Timestamp	java.sql.Timestamp
DECIMAL	java.lang.BigDecimal	java.math.BigDecimal
FLOAT	double	double
INT8	long	long
INTEGER	int	int
INTERVAL	com.informix.lang.Interval	java.lang.String
LIST	java.util.ArrayList	--- Not Supported ---
LVARCHAR	java.lang.String	java.lang.String
MONEY	java.lang.BigDecimal	java.math.BigDecimal
MULTISET	java.util.HashSet	--- Not Supported ---
NCHAR	java.lang.String	java.lang.String
NVARCHAR	java.lang.String	java.lang.String
ROW	java.sql.Struct	--- Not Supported ---
SERIAL	int	int
SERIAL8	long	long
SET	java.util.HashSet	--- Not Supported ---
SMALLFLOAT	float	float
SMALLINT	short	short
TEXT	java.io.InputStream	java.lang.Clob
UDT	java.sql.SQLInput, java.sql.SQLOutput	--- Not Supported ---
VARCHAR	java.lang.String	java.lang.String

Using an Application Server

It is very common to use Java for web applications. For this purpose you need an application server. IBM provides a free application server called WebSphere® Application Server Community Edition (WAS CE). It is also possible to get support for it if you decide to use it in a production environment. The current WAS CE release is version 2.1.0.1. You can find out more about WAS CE and how to download it at:

`http://www-01.ibm.com/software/webservers/appserv/community/`

Similarly to all the other IBM downloads, you will need to register before you can download the product. WAS CE is based on Apache Geronimo and Apache Tomcat. The download size is between 74MB and 154MB depending on if you decide to download the version that includes the IBM JDK 1.50.

The focus of this chapter is not to show you how to use an application server but, to get you started, here is some essential information that you will need.

The default installation directory, on Unix-type platforms is:

```
/opt/IBM/WebSphere/AppServerCommunityEdition
```

We'll refer to the installation directory as `$WASCE_HOME`.

Before you start WAS CE, you need to install the Informix JDBC driver in the application server. Do the following:

1. Go to the `$WASCE_HOME/repository/com` directory
2. Create the following directory: `informix/ifxjdbc/11.50`
3. Copy the `ifxjdbc.jar` file into `$WASCE_HOME/repository/con/Informix/ifxjdbc/11.50`
4. Rename `ifxjdbc.jar` to `ifxjdbc-11.50.jar`

To start WAS CE, you have to go to the `$WASCE_HOME/bin` directory and execute `startup.sh`. The administrator username is `system` and its password is `manager`. You can see the progress of the startup and error messages in `$WASCE_HOME/var/log/server.out`.

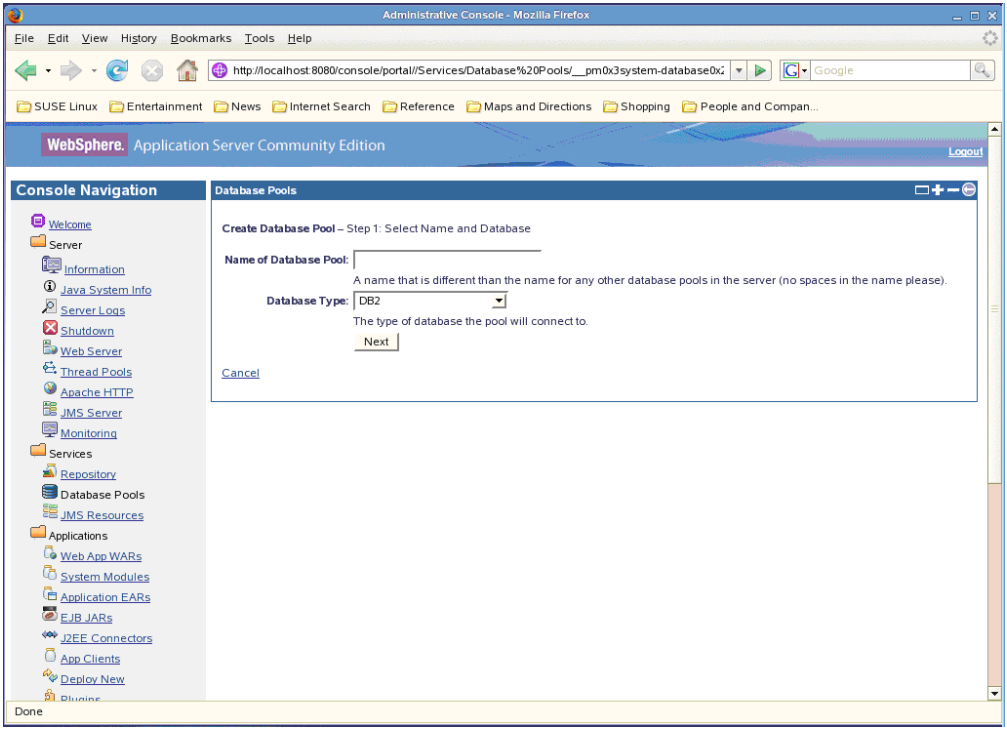
The server is started on port 8080. Assuming that you are starting it on the local machine, you can access the application server at the following URL:

```
http://localhost:8080
```

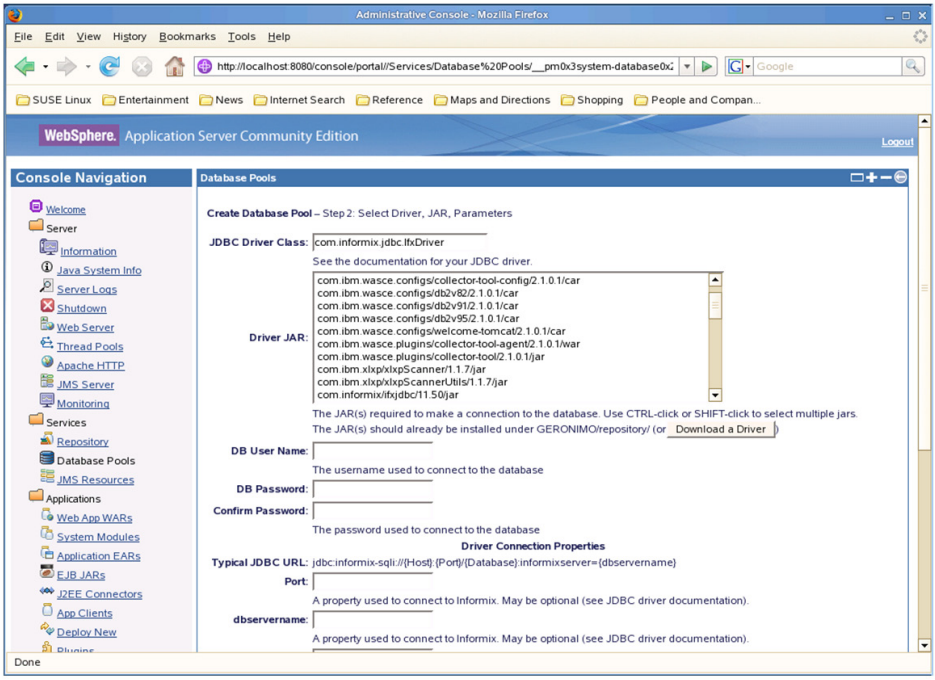
And the administrative console at:

```
http://localhost:8080/console
```

To access IDS through the application server, you have to go to the administrative console and define a data source by selecting Database Pools from the menu selection. Then click on "`Using the Geronimo database pool wizard`". At this point, the screen will look as follows:



To use the IBM Data Server driver, select the database type DB2. On the next screen, select the driver jar: `com.ibm.db2/db2jcc/9.5/jar`.



If you want to use the Informix JDBC driver, select the database type Informix. On the next screen, you then select the driver jar: `com.informix/ifxjdbc/11.50/jar`.

An application uses the data sources (database pools) defined in WAS CE. To tie the application with the server, you need to have a special file called `geronimo-web.xml` that is found under the directory `WEB-INF`. Assuming that an application called `DBAccessWeb` uses two data sources, one called `ids` and another one called `idsdrda`, the `geronimo-web.xml` file would look as follows:

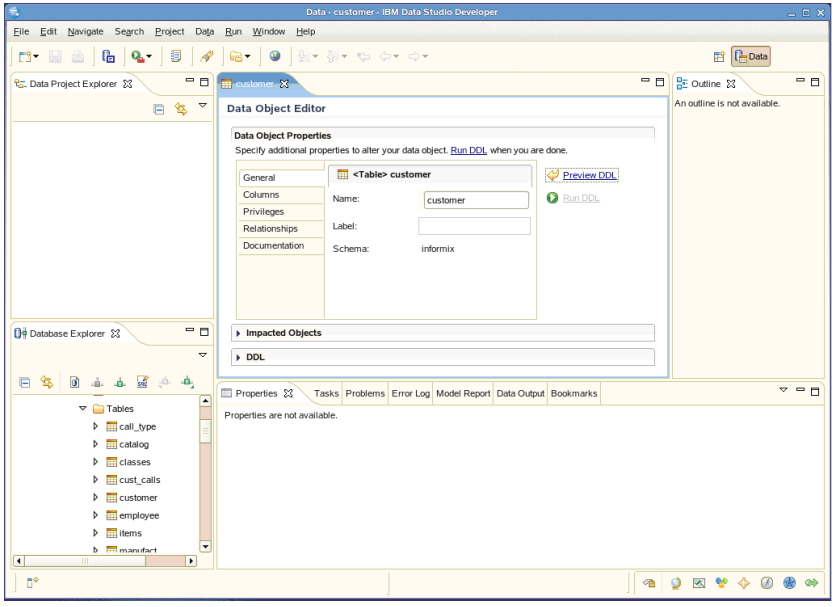
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://geronimo.apache.org/xml/ns/j2ee/web-1.1"
xmlns:naming="http://geronimo.apache.org/xml/ns/naming-1.1" xmlns:sec="http://
geronimo.apache.org/xml/ns/security-1.1" xmlns:sys="http://geronimo.apache.
org/xml/ns/deployment-1.1">
    <sys:environment>
        <sys:moduleId>
            <sys:groupId>DBAccessWeb</sys:groupId>
            <sys:artifactId>DBAccessWeb</sys:artifactId>
            <sys:version>1.1.1</sys:version>
            <sys:type>war</sys:type>
        </sys:moduleId>
        <sys:dependencies>
        <sys:dependency>
        <sys:groupId>console.dbpool</sys:groupId>
        <sys:artifactId>ids</sys:artifactId>
        </sys:dependency>
        <sys:dependency>
```

```
        <sys:groupId>console.dbpool</sys:groupId>
        <sys:artifactId>idsdrda</sys:artifactId>
        </sys:dependency>
        </sys:dependencies>
    </sys:environment>
    <context-root>/DBAccessWeb</context-root>
    <naming:resource-ref>
        <naming:ref-name>ids</naming:ref-name>
        <naming:resource-link>ids</naming:resource-link>
    </naming:resource-ref>
    <naming:resource-ref>
        <naming:ref-name>idsdrda</naming:ref-name>
        <naming:resource-link>idsdrda</naming:resource-link>
    </naming:resource-ref>
</web-app>
```

The key in this file is the definition of the database resource `ids` and `idsdrda`. Once this is provided, the application can access the database through the database pools defined in WAS CE.

Development Tools

Development tools are essential especially when it comes to writing web applications. IBM has many tools available for software development. You can download a trial version of IBM Data Studio Developer 2.1. This is an eclipse-based product that includes many features such as web services generation and Java code. It provides a solid foundation for Java development and includes many features that make it sassier to manipulate objects in the database such as creating tables, looking at the content of a table, and so on. The following screen shows the data perspective in IBM Data Studio:



You can download IBM Data Studio Developer 2.1 at:
`http://www-01.ibm.com/software/data/studio/developer/`

Reference Material

- *IBM Informix JDBC Driver Programmer's Guide, Version 3.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-9421-00
- *IBM Data Server Driver for JDBC and SQLJ for IDS, v3.52*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-9534-00
- *IBM Informix Guide to SQL: Syntax, Version 11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-7751-01
- *IBM Informix Guide to SQL: Reference, Version 11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-7750-00#
- *IBM Informix Guide to SQL: Tutorial, Version 3.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-9432-01#

5

Informix and PHP

Informix Dynamic Server (IDS) supports PHP through the PHP Data Object (PDO) standard interface. There are two drivers available: The Informix-specific driver and the IBM Data Server driver.

The Informix-specific driver is available as an open-source driver on the PHP pecl[3] site. The IBM Data Server driver is available as part of the IBM Data Server Client package in binary format and on the pecl site in source format.

This chapter discusses which driver to use and how to set up the development environment for each driver.

Picking a Driver

IBM's long term direction is to use the IBM Data Server driver but your environment will determine which driver to use. For example, if the Data Server driver is not available on the platform you are using, then you have to use the Informix driver. Another reason to pick the Informix driver over the Data Server driver is the data types that must be supported by your application. As mentioned in Chapter 3, some data types are not supported by the Data Server driver (since it depends on the Data Server Client software development kit (SDK).

If you are working exclusively with IDS, there is no problem with using the Informix-specific driver. If you are working in an environment where you also use the IBM DB2 database server, you will either have to use the IBM Data Server driver or use both drivers simultaneously.

The following discussion starts with the Informix-specific driver because you should strongly consider using the Open Admin Tool (OAT) for system administration as discussed below.

Installing the Informix-Specific Driver

One assumption here is that PHP is used for web applications through the use of a web server. Creating the entire environment from scratch is beyond the scope of this chapter. For more information on how to install and configure the entire environment, please see the developerWorks® articles listed in the reference material section at the end of the chapter.

Using OAT as your Web Server

An easy way to set up your environment is to use the Open Admin Tool (OAT) for IDS. OAT is an open-source, PHP-based, administration tool for IDS. Even though OAT is open-source, it is maintained by a team of engineers from the IBM Informix lab. OAT gives you an easy way to perform database administration through a GUI interface. For example, you can easily monitor the disk space usage. Figure 1 shows an example of that:

3. The PHP pecl site is a repository of extensions to PHP such as database drivers.

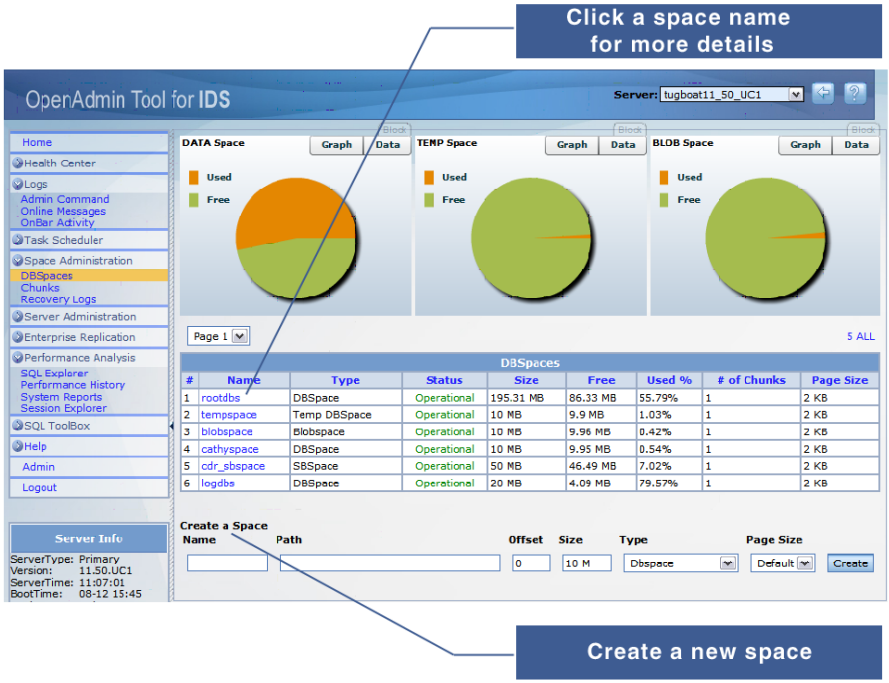


Figure 1: Database Space Usage Information

By installing this product, you get a ready-to-go PHP environment and you also have an easy-to-use tool to do monitoring and administration of the database server. If you intend on using the IBM Data Server driver, you can always add it after OAT is installed. At the time of this writing, the OAT product release is version 2.22 released in August 2008. It includes:

- IBM Informix I-Connect 3.50
- Apache 2.2.4
- PHP 5.2.4
- PDO_INFORMIX 1.1.0

The I-Connect product is the runtime version of the Informix Client SDK discussed in Chapter 3.

The OAT product is available as an automated install for Windows, Linux, and Macintosh OS X (64-bit only). You can download the product starting from the Open Admin Tool Web site at:

`http://www.openadmintool.org`

This will lead to the IBM site and will require you to register and login. The download is around 120MB in size. Once you have completed the download, the installation is as follows:

1. Start the installation with one of the following methods:

   - GUI Mode: launch the installation executable:
     Windows: `install.exe`
     Linux: `install.bin`
     MAC OS X: unzip the `install.zip file`, and then launch the install.app file

- Console Mode: enter the following commands:
  Windows: `install.exe -i console`
  Linux: `install.bin -i console`

2. Accept the license agreement to continue.
3. Select the installation directory; the default is:
   For Windows: `C:\Program Files\OpenAdmin`
   For Linux: `/opt/OpenAdmin`
   For MAC OS X: `/Applications/OpenAdmin/`
   Click Next.
4. Choose an available port number for the Web server.
5. For Windows users: provide an Apache service name.
6. Specify the Host Name, the name of the computer where the database server is located.
7. Click Next. The Security Features page is displayed.
8. To enable password-protection for the OpenAdmin Tool administration pages, select the check box and click Next. The OAT Administrator login setup page is displayed.
9. Enter a user name and password and click Next. The Plug-in page is displayed.
10. Ensure that the plug-ins that you want to install are selected and click Next. License information for each selected plug-in is displayed.
11. Click "I accept the terms of the License Agreement" for each plug-in and then click Next. The Pre-Installation Summary page is displayed.
12. Review your selections and click Install to continue with the installation of Open Admin Tool. When the installation is complete, the following message is displayed:
    "OpenAdmin Tool has been installed successfully. Please visit http://servername:portnumber/openadmin/ to use the OpenAdmin Tool": Where:
    - *servername* is the name of the machine where the Web server is running. This can be localhost on Windows.
    - *portnumber* is the port number that you provided in step 4. For example: http://localhost:portnumber/openadmin/index.php
13. Click Done.
    - For Linux and MAC OS X: The OpenAdmin Tool configuration page opens in your default Web browser.
    - For Windows: Your machine needs to reboot. Once your desktop is restored, go to the Start menu and click "OpenAdmin Tool for IDS".

With the download of the product, you can also download a README file and the release notice of the product.

Since the Informix PDO driver included with OAT is a little old, it is strongly suggested that you upgrade the driver to the latest version. This is discussed in the next section.

Installing the PDO_INFORMIX driver

If you already have a Web server that includes PHP, you can download and install the PDO_INFORMIX driver. It is available on the PHP pecl site at:

`http://pecl.php.net/package/PDO_INFORMIX`

Note that despite being an open source driver, it is maintained by an engineer from the Informix lab. The latest driver is the `PDO_INFORMIX-1.2.5.tgz`. You can extract the content with the following command:

```
$ tar zxvf PDO_INFORMIX-1.2.5.tgz
```

To install the PDO_INFORMIX driver, you need to compile the code using the Informix client SDK. In the following example, we are assuming that the client SDK is installed in `$INFORMIXDIR` and that a "C" compiler is available. A `pecl` extension can be prepared for compilation using the `phpize` command followed by other commands as shown below:

```
$ cd PDO_INFORMIX-1.2.5
$ phpize
Configuring for:
PHP Api Version:              20041225
Zend Module Api No:           20050922
Zend Extension Api No: 220051025
$ ./configure –with-pdo-informix=$INFORMIXDIR
. . . about 100 lines of messages . . .
$ make
. . . multiple messages . . .
$ make install
```

Depending on the platform you are on, you may have to download additional open-source software, such as `autoconf`, to be able to go through the `phpize` command.

On successful completion, the shared library `pdo_informix.so` will be put in the PHP extensions directory. You will need to modify the `php.ini` file to add the line "`extension=pdo_informix.so`" or add an entry in the PHP configuration directory.

The easiest way to find out what you need to modify is to look at the setup of your Web server. Create the following script as `phpinfo.php` in the `htdocs` directory:

```
<?php
phpinfo();
?>
```

The execution of the script will give you a lot of information. What you are looking for is in the first visible section of the output as shown in figure 2.

PHP Version 5.2.5

System	Linux suse1 2.6.16.60-0.21-smp #1 SMP Tue May 6 12:41:02 UTC 2008 i686
Build Date	Apr 23 2008 23:25:46
Configure Command	'../configure' '--prefix=/usr' '--datadir=/usr/share/php5' '--mandir=/usr/share/man' '--bindir=/usr/bin' '--with-libdir=lib' '--includedir=/usr/include' '--sysconfdir=/etc/php5/apache2' '--with-config-file-path=/etc/php5/apache2' '--with-config-file-scan-dir=/etc/php5/conf.d' '--enable-libxml' '--enable-session' '--with-mm' '--with-pcre-regex' '--enable-xml' '--enable-simplexml' '--enable-spl' '--enable-filter' '--disable-debug' '--enable-inline-optimization' '--disable-rpath' '--disable-static' '--enable-shared' '--program-suffix=5' '--with-pic' '--with-apxs2=/usr/sbin/apxs2' '--disable-all' '--disable-cli'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php5/apache2
Loaded Configuration File	/etc/php5/apache2/php.ini
Scan this dir for additional .ini files	/etc/php5/conf.d
additional .ini files parsed	/etc/php5/conf.d/ctype.ini, /etc/php5/conf.d/dom.ini, /etc/php5/conf.d/gd.ini, /etc/php5/conf.d/iconv.ini, /etc/php5/conf.d/json.ini, /etc/php5/conf.d/pdo.ini,

Figure 2: web Server PHP configuration

The two entries of interest are "`Loaded Configuration File`" and "`Scan this dir for additional .ini files`". You can either add an entry in the `php.ini` file or add a file in the `conf.d` directory `(pdo_informix.ini)`. Both the file content and the entry have the same format: `extension=pdo_informix.so`

After re-starting the Web server, you should execute `phpinfo.php` and look for the PDO information to make sure the Informix driver was loaded. This is shown in Figure 3.

PDO

PDO support	enabled
PDO drivers	informix, sqlite, sqlite2

pdo_informix

pdo_informix support	enabled

pdo_sqlite

PDO Driver for SQLite 3.x	enabled
PECL Module version	(bundled) 1.0.1 $Id: pdo_sqlite.c,v 1.10.2.6.2.2 2007/03/23 14:30:00 wez Exp $
SQLite Library	3.3.17

Figure 3: PDO Information

In this case, we see that three PDO drivers have been loaded in the server. If the Informix driver is not listed, you need to review your installation. It could be that the driver (`pdo_informix.so`) was not installed in the proper location.

Trying the PDO_INFORMIX Driver

You can test the PDO_INFORMIX connectivity with the Informix IDS server with the following program:

```
<?php
  $informixdir = getenv("INFORMIXDIR");
  $uname = "informix";
  $password= "password";
  $conn_string = "informix: host=jroix; service=9088;
database=stores; server=demo_on; protocol=onsoctcp;
TRANSLATIONDLL=$informixdir/lib/esql/igo4a304.so ;
CLIENT_LOCALE=en_us.8859-1 ; DB_LOCALE=en_us.8859-1 ";
  $sql = "SELECT dbinfo('version', 'major') version FROM systables WHERE
tabid=1";
  try {
  print "<li>informixdir = $informixdir</li>\n";
  $conn = new PDO($conn_string, $uname, $password);
  $conn->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
print "<li>Got a connection</li>\n";
  $stmt = $conn->query($sql);
  if ( ! $stmt ) {
  print "Error in execute: stmt->execute()\n";
  print "errInfo[0]=>$err[0]\nerrInfo[1]=>$err[1]\nerrInfo[2]=>$err
  [2]\n";
    }
    $row = $stmt->fetch();
    print "<li>IDS version " . $row['VERSION'] . "</li>\n";
```

```
  } catch (Exception $e) {
        print "Exception messsage: {$e->getMessage()}\n";
         exit(0);
  }
?>
```

After setting the proper hostname in the `conn_string` variable and the proper username and password, the program should run as is. It should return something similar to the following:

- `informixdir = /opt/IBM/informix`
- `Got a connection`
- `IDS version 11`

At this point, you know that your environment is set properly and you are ready to start using PHP with IDS.

Installing the IBM Data Server Driver

Most people use PHP for web applications. This requires the use of a web server, usually Apache, a PHP processor, and the appropriate database drivers. The easy way to get going with the IBM Data Server Driver for PHP (`PDO_IBM`) is to use the free binary packaging provided by Zend called "Zend Core for IBM".

Zend Core for IBM

This product includes all you need to run PHP applications that access IDS. It is currently available for AIX, Linux, and Windows. At the time of this writing, the current version of Zend Core for IBM is 2.0. You can find more information and how to download it at:

`http://www.zend.com/en/products/core/for-ibm`

The download is 62MB in size on Linux and 81MB on Windows. The installation requires answering simple questions. Selecting the defaults is usually the best approach. The default installation directory is:

- On Linux: `/usr/local/Zend/Core`
- On Windows: `C:\Program Files\Zend\Core for IBM`

Zend Core installation includes an administrative console. During the installation you will need to provide a password to access the console through the web server. You can access the administrative console at:
`http://<hostname>:<portNumber>/ZendCore`
Where the hostname is your machine name or localhost, and the port number is the number you entered during the installation, 80 being the default.

Zend Core for IBM comes with both the `PDO_INFORMIX` and the `PDO_IBM` drivers. Before you can use the IBM Data Server driver (`PDO_IBM`) to access IDS, you must disable the `PDO_INFORMIX` driver through the administrative console. You have to access the section "Configuration" and the subsection "Extensions" in the menu at the top of the console display. Once there, you need to scroll down about half way down to see the `PDO_IBM` and `PDO_INFORMIX` driver entries as shown in Figure 4.

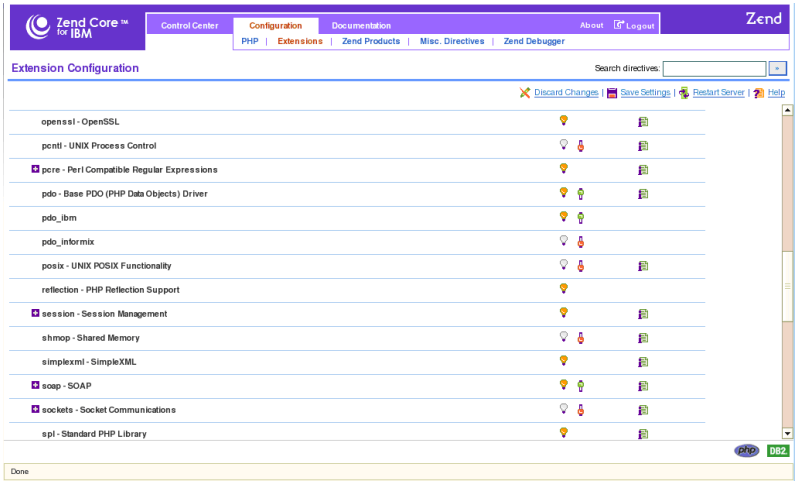


Figure 4: Zend Core Administrative Console

To disable `PDO_INFORMIX`, you must click on the vertical scrollbar next to the light bulb on the line for `pdo_informix`. You must then save the settings by clicking on the "Save Settings" option located in the upper-right part of the screen, and restart the server by clicking on "Restart Server" located in the upper-right part of the screen.

If you wanted to use the `PDO_INFORMIX` driver with Zend Core for IBM, you will need to disable the `pdo_ibm` driver. Note that you can use both drivers simultaneously if you install the drivers yourself in an existing installation. For example, you could install the PDO_IBM driver in the OAT installation and be able to use both drivers.

Installing the Driver in an Existing Installation

To install the driver in an existing installation, you need to install the IBM Data Server Client. Please refer to Chapter 3 for more installation information.

Once the Client product is installed, you have two possibilities for your installation. The first one is to use the already compiled driver or to start from source code and compile the code.

Using the Binary Driver

The binary driver is located in the data server client (assuming `DB2_HOME`) under the dsdriver directory. For example, on 32-bit Linux, the driver would be:

```
$DB2_HOME/dsdriver/php32/pdo_ibm_5.2.1.so
```

You need to copy this file into the PHP extension directory as `pdo_ibm.so`. As we saw in the "Installing the PDO_INFORMIX Driver" section above, you need to make sure you know where the extensions directory is (for example: `/usr/lib/php5/extensions`) and where the `php.ini` file is. After copying the file in the extensions directory, you need to modify the `php.ini` file or add a file to the configuration directory (`ex: /etc/php5/conf.d`). Both methods add the following content:

```
extension=pdo_ibm.so
```

After restarting your web server, you should check if the new PDO driver has been loaded. This can be done by requesting the `phpinfo()` information through the Zend administrative console.

Installing the Driver from Source

To install the driver from the source, you follow the same approach as what we have seen above in the section "Installing the PDO_INFORMIX Driver". You can find the PDO_IBM driver on the pecl site at:

```
http://pecl.php.net/package/PDO_IBM
```

The current release at the time of this writing is version 1.2.5. Execute the following command to extract the content of the archive:

```
$ tar xzvf PDO_IBM-1.2.5.tgz
```

This will create a directory named `PDO_IBM-1.2.5`. Assuming that the data server client is installed in `$DB2_HOME`, you need to do the following to compile and install the driver:

```
$ cd PDO_IBM-1.2.5
$ phpize
Configuring for:
PHP Api Version:              20041225
Zend Module Api No:           20060613
Zend Extension Api No: 220060519
$ ./configure --with-pdo-ibm=$DB2_HOME
. . .about 100 lines . . .
$ make
. . . multiple messages . . .
$ make install
```

Make sure `pdo_ibm.so` is copied to the appropriate PHP extensions directory and then add the appropriate entry in `php.ini` or a new file in the `conf.d` directory as mentioned in the previous section.

As mentioned earlier, you need to restart your web server and verify that the `PDO_IBM` driver is now recognized.

Trying the PDO_IBM Driver

You can test the connectivity to IDS with the PDO_IBM driver with the following program:

```
<?php
  $uname = "informix";
  $password= "password";
  $conn_string = "ibm:HOSTNAME=jroix;PORT=9089;DATABASE=stores;
protocol=TCPIP;";


  $sql = "SELECT dbinfo('version', 'major') version
          FROM systables WHERE tabid=1";


  try {
    $conn = new PDO($conn_string, $uname, $password);
```

```
    $conn->setAttribute(PDO::ATTR_ERRMODE,
  PDO::ERRMODE_EXCEPTION);
  print "<li>Got a connection</li>\n";
  $stmt = $conn->query($sql);
  if ( ! $stmt ) {
    print "Error in execute: stmt->execute()\n";
    print
"errInfo[0]=>$err[0]\nerrInfo[1]=>$err[1]\nerrInfo[2]=>$err
[2]\n";
    }
    $row = $stmt->fetch();
    print "<li>IDS version " . $row['VERSION'] . "</li>\n";
  } catch (Exception $e) {
    print "Exception messsage: {$e->getMessage()}\n";
    exit(0);

  }
?>
```

The program will output the following in your browser window:

- `Got a connection`
- `IDS version 11`

You are now ready to use PHP to access IDS.

Reference Material

- *Informix Product Family – PHP Application Development web page*
  http://www-01.ibm.com/software/data/informix/php/
- *A Step-By-Step How-To Guide to Install, Configure, and Test a Linux, Apache, Informix, and PHP Server*
  http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0606bombardier/
- *A Step-By-Step How-To Guide to Install, Configure, and Test a Windows, Apache, Informix, and PHP Server*
  http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0607bombardier/
- *Informix Functions (PDO_INFORMIX)*
  http://kr2.php.net/manual/en/ref.pdo-informix.php
- *IBM Functions (PDO_IBM)*
  http://kr2.php.net/manual/en/ref.pdo-ibm.php

6

Ruby and Rails

Ruby is an object-oriented scripting language that has gained in popularity in recent years. One of the reasons for this popularity is the Rails framework that provides a model-view-controller (MVC) organization for web applications.

This chapter discusses the use of Ruby and Rails to access IBM Informix Dynamic Server (IDS). The installation was tested using Ruby 1.8.6 and Rails 2.1.2.

Drivers Choices

As mentioned before, there are two communication paths to IDS. One is through the "traditional" Informix communication called sqli and the other one is through a DRDA connection. The long term direction is with the IBM Data Server Client drivers using DRDA connectivity.

The IBM Data Server driver does not support all the data types that are included in IDS. Please see Chapter 3 for more details. You may not need to use these data types so you may still be able to use the IBM Data Server driver. If you need to use types such as user-defined types in your applications, you will need to use Informix-specific drivers.

In the Ruby and Rails environment, the only supported driver is the IBM Data Server driver. The Informix drivers (one for Ruby and one for Rails, are available on RubyForge as open-source products. Despite not having official support, the RubyForge project administrator is responsive to problems.

The bottom line is that, if possible, you should use the IBM Data Server driver. This chapter provides information on how to setup the environment for each driver.

Getting the IBM Data Server Driver

The Ruby and Rails driver is available in binary format as part of the IBM Data Server Client. At the time of this writing, the available driver is ibm_db-0.8.5. You can get a more recent version by getting the driver from the RubyForge web site at:

`http://rubyforge.org/projects/rubyibm`

The latest driver is version 1.0.0, released on November 6, 2008. You can download `ibm_db-1.0.0.gem` file for all platforms or `ibm_db-1.0.0.mswin32.gem` for Windows i386-type platforms.

This latest driver comes in source format and requires compilation during the installation. It is the same driver that comes with the IBM Data Server Client and is maintained by IBM engineers. Before you can install the driver, you need to install the IBM Data Server Client since it requires it for the compilation. Refer to Chapter 3 to learn how to install the IBM Data Server Client.

Installing the IBM Data Server Driver

The installation of the IBM Data Server Driver requires two steps: The installation of the gem file and the modification of `active_record.rb` (if you are using Rails version before 2.0).

The installation of the gem file that comes with the IBM Data Server Client is as follows:

```
# gem install ibm_db-0.8.5.gem
Successfully installed ibm_db-0.8.5
1 gem installed
Installing ri documentation for ibm_db-0.8.5…
Installing Rdoc documentation for ibm_db-0.8.5....
#
```

The ibm_db-1.0.0-mswin32.gem comes with the code already compiled so if you get the latest one from RubyForge, the installation will look as follows:

```
C:\>gem install ibm_db-1.0.0-mswin32.gem
Successfully installed ibm_db, version 1.0.0
Installing ri documentation for ibm_db-1.0.0-mswin32...
Installing RDoc documentation for ibm_db-1.0.0-mswin32...
```

On other platforms, running the gem installation from the source code gem file (`ibm_db-1.0.0.gem`) from RubyForge, you get the following:

```
# . /home/db2inst1/sqllib/db2profile
# export IBM_DB_DIR=/home/db2inst1/sqllib
# export IBM_DB_LIB=/home/db2inst1/sqllib/lib32
# gem install ibm_db-1.0.0.gem
Building native extensions.  This could take a while…
Successfully installed ibm_db-1.0.0
1 gem installed
Installing ri documentation for ibm_db-1.0.0…
Installing Rdoc documentation for ibm_db-1.0.0...
#
```

If you are using a version of Rails before version 2.0, you need to edit

```
<rubyhome>/gems/1.8/gems/activerecord-1.15.6/lib/active_record.rb
```

and add `ibm_db` to the list of available adapters for `RAILS_CONNECTION_ADAPTERS`. The entry would look something like this:

```
RAILS_CONNECTION_ADAPTERS = %w( mysql postgresql sqlite
firebird sqlserver db2 oracle sybase openbase frontbase
ibm_db )
```

At this point you are ready to try connecting to IDS.

Testing the IBM Data Server Driver Connectivity

The following Ruby script is a short test for connectivity with IDS:

```
require 'rubygems'
require 'ibm_db'


conn = IBM_DB::connect "DRIVER={IBM DB2 ODBC DRIVER};\
                        DATABASE=stores;\
                        HOSTNAME=jroix;\
                        PORT=9089;\
                        PROTOCOL=TCPIP;\
                        UID=informix;\
                        PWD=password;", '', '', nil
stmt = IBM_DB::exec conn, 'select * from customer'

IBM_DB::fetch_assoc stmt
```

The first line, `require 'rubygems'`, is not always required but does not interfere with the test. The important part of this program is the definition of a database connection starting on line 3. The connection information assumes the default Informix database server installation that is covered in Chapter 2. You will need to adjust the values for the `HOSTNAME` on line 6, UID on line 9, and `PWD` on line 10.

Assuming that the script name is `connect2ids.rb`, you can execute it using interactive Ruby with the following command:

```
$ irb  < connect2ids.rb
```

As part of the execution you will see result from getting a connection, a statement, and at the end, a display of the first row of the result set. From the moment we get a connection from the connect statement (lines 3 to 10), we know that we are able to talk to IDS. Here is a display of what you should see at the end of the execution:

```
irb (main):011:0> stmt = IBM_DB::exec conn, 'select * from customer'
=> #<IBM_DB::Statement:0xb7ab1640>
irb (main):013:0> IBM_DB::fetch_assoc stmt
=> { "city" => "Sunnyvale    " , . . .}
```

The next step is to make sure we can also talk to the IDS database using the Rails active record interface. We can test it using the following program:

```
require 'rubygems'
require 'ibm_db'
require 'active_record'

ActiveRecord::Base.establish_connection(
  :adapter => "ibm_db", :host => "jroix",
  :port => "9089", :database => "stores",
  :username => "informix", :password => "informix")

class Customer < ActiveRecord::Base
  set_table_name "customer"
  self.primary_key = "customer_num"
end

customer = Customer.find(101)
```

The key to accessing IDS from an active record is to provide the connectivity information as shown on the line starting with:
`ActiveRecord::Base.establish_connection`. You will need to put your own values for these parameters.

This program then creates a class called "Customer". For a class named Customer, Rails assumes that the table name is then class name, plural. This means that the default table name is "customers". The class definition overwrites the table name and overwrites the name of the column that represents the primary key.

We can execute the program, called `arec.rb`, using the following command:

```
$ irb < arec.rb
```

The program will execute and the `Customer.find(101)` line returns the appropriate row in the customer variable:

```
customer = Customer.find(101)
#<Customer customer_num: 101, fname: "Ludwig         ",
lname: "Pauli          ", company: "All Sports Supplies ",
address1: "213 Erstwild Court  ", address2: nil, city:
"Sunnyvale      ", state: "CA", zipcode: "94086", phone:
"408-789-8075      ">
```

You are now ready to start your Rails development.

Getting the Informix Driver

If you decide to use the Informix drivers instead of the IBM Data Server driver, you have to get one driver for Ruby and an adapter for Rails. You can get them at:

- Ruby: `http://ruby-informix.rubyforge.org`
  The latest release is version 0.7.1, available since April 2, 2008
- Rails: `http://rubyforge.org/projects/rails-informix`
  The latest release is version 1.1.0, available since December 5, 2007.

Remember that these drivers are not supported by IBM.

Installing the Informix Drivers

The installation of the drivers may require two steps: installing the Ruby-Informix driver and installing the Rails-Informix driver. The installation of the Ruby-Informix driver is as follows:

```
# gem install ruby-informix-0.7.1.gem
Building native extensions.  This could take a while…
Successfully installed ruby-informix-0.7.1
1 gem installed
Installing ri documentation for ruby-informix-0.7.1…
Installing Rdoc documentation for ruby-informix-0.7.1...
#
```

For the Windows platform, you can install the Ruby-Informix-0.7.1-i386-mswin32.gem gem file without requiring a compilation. The installation looks as follows:

```
C:\>gem install ruby-informix-0.7.1-i386-mswin32.gem
Successfully installed ruby-informix, version 0.7.1
Installing ri documentation for ruby-informix-0.7.1-i386-
mswin32...
Installing RDoc documentation for ruby-informix-0.7.1-i386-
mswin32...
```

The next step is to install the Rails-Informix adapter. This consists of extracting the files from the archive and copying informix_adapter.rb to the appropriate location:

```
# tar zxvf rails-informix-1.1.0.tar.gz
. . .
# cd rails-informix-1.1.0
# cp informix_adapter.rb /usr/local/lib/ruby/gems/1.8/gems/activerecord-
2.1.2
lib/active_record/connection_adapters
```

On Windows, you can extract the file form the rails-informix-1.1.0.zip and copy it to the appropriate location. For example, if Ruby was installed in `C:\ruby`, the location is: C:`\ruby\lib\ruby\gems\1.8\gems\activerecord-2.1.2\lib\active_record\connection_adapters`.

If you are using a version of Rails before 2.0, you also need to edit `<rubyhome>/gems/1.8/gems/activerecord-1.15.6/lib/active_record.rb` and add `informix` to the list of available adapters for `RAILS_CONNECTION_ADAPTERS`. The entry would look something like this:

```
RAILS_CONNECTION_ADAPTERS = %w( mysql postgresql sqlite
firebird sqlserver db2 oracle sybase openbase frontbase  informix )
```

Testing the Informix Driver

The Informix driver depends on the standard IDS connectivity definition. On Unix platforms, that means having the proper definition for:

- `INFORMIXDIR` environment variable
- `INFORMIXSERVER` environment variable
- `Sqlhosts` file that defines the available database server connections

On Windows systems, these definitions are kept in the registry. A database server definition can be created with the setnet32 program. Refer to Chapter 2 (IDS installation) and Chapter 3 (stand-alone clients installation) for more information. If you have followed the instruction in these chapters, your environment should be set up properly.

The following Ruby script allows you to test the connectivity to IDS:

```
require 'rubygems'
require 'informix'
conn = Informix.connect('stores', 'informix', 'password')
cur = conn.cursor('select * from customer')
cur.open.fetch
```

The `Informix.connect` command takes a database name, username, and password as arguments. If it returns a database connection, everything is working fine. The program goes one step further by executing an SQL statement and returning the first row of the result set. Assuming that the name of the script is `connect2ids2.rb`, you can execute it using the interactive Ruby program as follows:

```
$ irb < connect2ids2.rb
```

The following shows an example of the output of the last command of the script (fetch):

```
irb (main):005:0> cur.open.fetch
[101, "Ludwig    ", . . .]
```

The next step is to test the connectivity for the Rails framework using the active record. The following script provides this testing:

```
require 'rubygems'
require 'informix'
require 'active_record'

ActiveRecord::Base.establish_connection(
  :adapter => "informix", :host => "jroy_ix",
  :port => "9088", :database => "stores",
  :username => "informix", :password => "password")
```

```
class Customer < ActiveRecord::Base
  set_table_name "customer"
  self.primary_key = "customer_num"
end

customer = Customer.find(101)
```

This program then creates a class called "Customer". For a class named Customer, Rails assumes that the table name is then class name, plural. This means that the default table name is "customers". The class definition overwrites the table name and overwrites the name of the column that represents the primary key.

We can execute the program, named `arec2.rb`, using the following command:

```
$ irb <arec2.rb
```

The program will execute and the `Customer.find(101)` line returns the appropriate row in the customer variable:

```
customer = Customer.find(101)
#<Customer customer_num: 101, fname: "Ludwig         ",
lname: "Pauli          ", company: "All Sports Supplies ",
address1: "213 Erstwild Court  ", address2: nil, city: "Sunnyvale      ",
state: "CA", zipcode: "94086", phone: "408-789-8075      ">
```

You are now ready to start your Rails development.

Reference Material

- *IBM_DB Driver Documentation*
  http://rubyibm.rubyforge.org/docs/driver/0.9.0/rdoc/
- *IBM_DB Rails Adapter Documentation*
  http://rubyforge.org/docman/?group_id=2361
- *Ruby Library for IBM Informix*
  http://ruby-informix.rubyforge.org/doc/index.html
- *IBM Informix Guide to SQL: Syntax, Version 11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-7751-01
- *IBM Informix Guide to SQL: Reference, Version 11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-7750-00#
- *IBM Informix Guide to SQL: Tutorial, Version 3.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-9432-01#

7

.NET Environment

Nobody can talk about application development without mentioning the Microsoft .NET environment. In this chapter, we look at how we can use Informix Dynamic Server (IDS) in this environment.

IDS also supports an ODBC and OLE DB drivers for legacy windows environment. This chapter focuses on .NET since this is the stated Microsoft direction.

Drivers for .NET

The .NET environment provides multiple ways to access a database. The basic choices are to use an ADO.NET managed provider bridge to an ODBC or an OLE DB driver or to use a managed provider that can access the database directly without bridging to code outside the .NET framework.

IDS supports two different .NET providers: The IBM Data Server .NET provider and the Informix .NET provider. The IBM long term direction is to use the Data Server .NET provider but there may be cases where you may have to use the Informix . NET provider.

The IBM Data Server .NET provider does not support all the IDS features. Take a look at Chapter 3 for additional details. If you don't know which provider to use, start with the IBM Data Server .NET provider.

The rest of this chapter discusses the use of both drivers, starting with the IBM Data Server .NET provider.

IBM Data Server .NET Provider

The IBM Data Server .NET provider is a separate product from IDS. Before you can proceed, you must install the IBM Data Server client. This is described in Chapter 3. Once you have completed the installation, you can resume reading this chapter.

The latest version of the Data Server Client supports .NET framework Version 3.0. It also comes with an add-in for Visual Studio 2008. Previous versions of Visual Studio (2005, 2003) are also supported. The add-in is installed as part of the installation of the Data Server client.

The IBM Database Add-In for Visual Studio includes many features that can be used with an Informix database. These include:

- **IBM Database Project types**
  The Add-Ins introduce a new IBM Projects folder that includes an IBM Database Project type for developing IBM data server scripts.
- **Server Explorer**
  The Add-Ins extend the Server Explorer tool window in the Visual Studio environment. They provide Visual Studio users with access to IBM data connections.
- **IDS SPL Editor**
  The Add-Ins also include an IDS (Informix Dynamic Server) SPL editor. With the editor you can change and view the code in your IDS procedures, functions, objects and scripts.

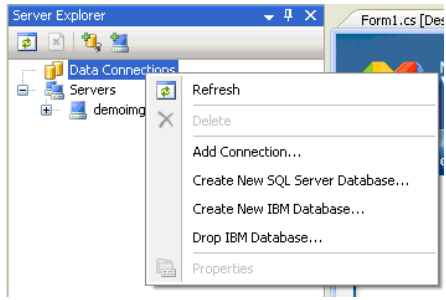
The IBM Database Add-In for Visual Studio integrates IntelliSense features and help for the different features within Visual Studio. For more information on the IBM Database Add-In for Visual Studio, please refer to the following URL:

http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/topic/com.ibm.swg.vs.addins.doc/html/ibmdevori-DevAddInOverview.htm

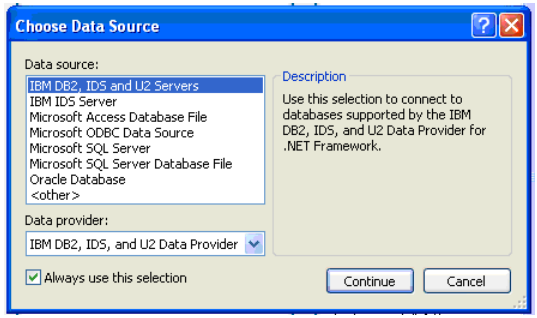
Connecting to IDS

The first step is likely to create a connection in the Server Explorer so we can look at what is available in the database. If you want to create a connection through the server explorer, you must go through the following steps:

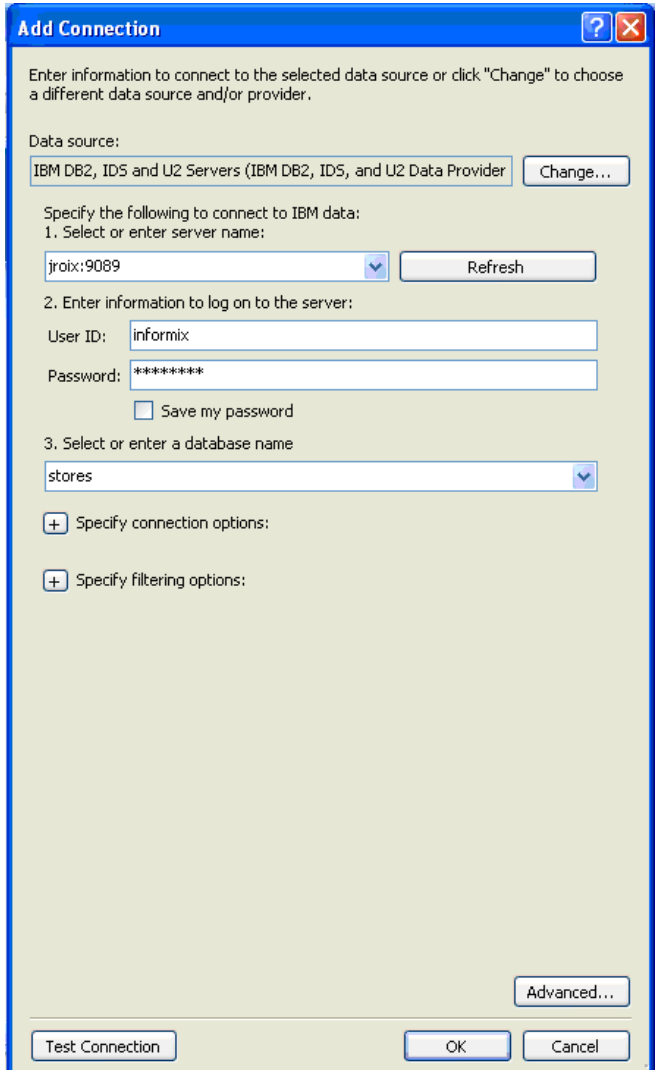
1) Open the Server Explorer, right-click on Data Connection. Then, select “Add Connection…”



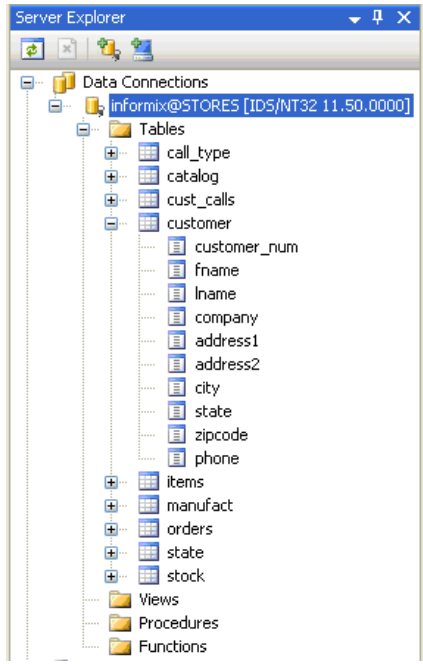
What pops up is a window that asks you to select the data source. In our case, we want the first selection that is the IBM DB2, IDS and U2 Servers.



By pressing Continue, you get the following window:



Note that the server name includes the name of the server followed by a semi colon and the port number the database server is listening to. You can use test connection to insure you entered the proper information. By clicking OK, the connection is created in the Server Explorer. You can then expand the different levels of the database connection to see the following:



From there you can manipulate the tables as expected through the Server Explorer.

You can establish a connection through drag-and-drop using the classes provided in Visual Studio. To better understand what is happening, we'll look at some code that relates the connection information we have seen earlier.

You can establish a connection to IDS using a generic approach. Here is a slightly modified example coming from the manual "*DB2 version 9.5 for Linux, UNIX, and Windows Developing ADO.NET and OLE DB Applications*" listed in the reference section:

```
DbProviderFactory factory =
          DbProviderFactories.GetFactory("IBM.Data.DB2");
DbConnection conn = factory.CreateConnection();
DbConnectionStringBuilder sb =
          factory.CreateConnectionStringBuilder();

if( sb.ContainsKey( "Database" ) )
{
       sb.Remove( "database" );
       sb.Add( "database", "stores" );
}
conn.ConnectionString = sb.ConnectionString;
conn.Open();
```

You may have to add additional attributes for the connection. This could include a host name, port number the database server listens to, username and password as we'll see below.

You can also use the Data Server .NET provider classes directly. Here is a C# example that creates a connection. Note that the connection string includes more complete information on the location of the server:

```
String cs = "Server=jroix:9089;Database=stores;
UID=informix;PWD=password";

DB2Connection conn = new DB2Connection(connectString);
conn.Open();
```

Once you have a connection, you can access the database using IBM variations of standard classes. Here is a C# example that executes a SELECT statement:

```
DB2Command cmd = conn.CreateCommand();
DB2Transaction trans = conn.BeginTransaction();
cmd.Transaction = trans;
cmd.CommandText = "SELECT lname, fname " +
" FROM customer WHERE customer_num < 110";
DB2DataReader reader = cmd.ExecuteReader();
while(reader.Read())
{
lastname = reader.GetString(0);
firstname = reader.GetString(1);

// Console.WriteLine(lastname + ", " + firstname);
. . .
}
reader.Close();

trans.Commit();
```

With this information, you should be ready to use the IBM Data Server .NET provider with Informix.

Informix .NET Provider

If you are using .NET on the same machine where you installed IDS, you are ready to go. Otherwise, you must install the Informix client-SDK. Refer to Chapter 3 to find out where to download it from and how to install it.

The Informix .NET provider includes support for Visual Studio 2003 and 2005. Currently, Visual Studio 2008 is not supported.

The current version of the Informix .NET provider implements the ADO.NET 2.0 interface. Version 1.1 is supported in an older version of the Informix client SDK. The programming interfaces supported are:

- Visual Basic .NET
- Visual C# .NET
- Visual J# .NET
- ASP.NET

To install the Informix .NET provider, start Visual Studio and go to the solution explorer window:

- Click on References
- Click Browse and navigate to the assembly
  The file is located in the Informix installation directory, represented by `INFORMIXDIR`, at `%INFORMIXDIR%\bin\net20`
- Select the assembly and click open
- Click Ok

Before you can use the .NET provider, you must add information in the database server by executing the `cdotnet.sql` script in the `sysmaster` database as user Informix. This script is included with the

Informix Client SDK.

From the machine where the Informix server is installed, start a command window as Informix and execute the following command:

C:\...> dbaccess –e sysmaster %INFORMIXDIR%\etc\cdotnet

This command accesses the sysmaster database and executes the commands that are in the file given as second argument.

You can now use the ADO.NET interface. The main IBM Informix .NET Provider classes that provide data access are:

- `IfxConnection`—for connection to a database and management of transactions
- `IfxCommand`—for issuing SQL commands
- `IfxDataReader`—for reading a forward-only stream of data records
- `IfxTransaction`—for controlling transactions
- `IfxDataAdapter`—for pushing data into a dataset and for reconciling changes in a dataset with the database

The following .NET Provider classes let you develop provider-independent code when you use the .NET Provider for the 2.0 framework:

- `DbProviderFactory`
- `DbConnectionStringBuilder`
- `DbCommand`

Here is an example of code that connects to IDS:

```
IfxConnection conn=new IfxConnection("Host=jroyix;
Service=9088; Server=ol_ids_1150_1;User ID=informix;
password=password; Database=stores");
conn.Open();
```

The connection attributes should be self explanatory. You will have to adjust the Host, and `password` attributes. If you are accessing the server as it was installed in Chapter 2, the other attributes should be fine.

Here is an example of accessing the database taken from the documentation. We assume we got a connection from the code shown above:

```
IfxDataAdapter idap = new IfxDataAdapter(
                        "select * from customer",conn);
DataSet ds = new DataSet("customer");
idap.Fill(ds,"customer");
```

For more information, please consult the IBM Informix .NET provider Reference Guide version 3.5 listed in the reference section below.

Reference Material

- *DB2 version 9.5 for Linux, UNIX, and Windows*
  Developing ADO.NET and OLE DB Applications
  ftp://ftp.software.ibm.com/ps/products/db2/info/vr95/pdf/en_US/db2ane951.pdf
- *IBM Informix .NET Provider Reference Guide, v3.5*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-9425-00
- *IBM Informix ODBC Driver Programmer's Manual, v3.5*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-9423-00
- *IBM Informix OLE DB Provider Programmer's Guide, v 3.5*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-9424-00

A

Database Administration Basics

Any developer that uses a database system needs to know some basic things about the server. This appendix gives you the information you need to complete usual tasks. This includes:

- Starting and stopping IDS
- Checking the server status and disk utilization
- Basic understanding of logs
- Loading and unloading data

The Open Admin Tool for IDS (OAT)

There is a lot more that can be done than what is covered in this appendix. For example, you can look at the SQL statements that were executed and find out about their resource utilization. This can be a great way to find out that you should add an index on a specific column for example. OAT gives you an interface for that. It also gives you an easy to use interface to do most administrative functions. Some of its features include:

- Health center
- Logs management
- Task scheduler
- Server administration
- Performance analysis
- SQL toolbox
- Help

OAT is a PHP based open-source administration tool. You can find more information about it at:

`http://www.openadmintool.org`

We also discuss OAT installation as a way to setup a PHP environment in Chapter 5. OAT gives you an easy way to handle most administrative tasks. I strongly recommend that you use it.

Starting and Stopping IDS

This was discussed in Chapter 2, Installing IDS. On Windows, you have to go to the services screen (Control Panel→Administrative Tools→Services [Informix IDS…]). From there you can start, stop, and restart the database engine. You can also edit the service properties and set it to start automatically when the system is brought up. This is the default setting.

In UNIX-type systems, it is done through two commands. To start IDS, you execute the `oninit` command at a UNIX prompt as either user `root` or `informix`. It is possible to automate the starting and stopping of the database server by modifying the `/etc/rc.d` directory as described in Chapter 2.

To shutdown IDS, you use the `onmode` command: `onmode -ky`.

Checking the Server Status

There are a few basic things you want to check once in a while. The main ones are the database server status and disk space utilization. This can be done either through the OAT interface discussed earlier or through the `onstat` command.

The database server can be stopped, in recovery mode, or up. The command "`onstat –`" gives you this information. If the database server is not running, you receive an error:

```
$ onstat –
shared memory not initialized for INFORMIXSERVER 'demo_on'
```

If the server is up and running, you see something similar to the following:

```
$ onstat –

IBM Informix Dynamic Server Version 11.50.UC2DE -- On-Line --
Up 01:27:13 – 38208 Kbytes
```

The key to this message header is "-- On-Line --". This indicates that the database server is operational. If additional information is included, it could indicate a problem such as being blocked due to a long transaction.

To check for the disk space utilization, use the –d options:

```
$ onstat –d

IBM Informix Dynamic Server Version 11.50.UC2DE -- On-Line --
Up 00:10:29 -- 38208 Kbytes


Dbspaces
address         number   flags      fchunk   nchunks  pgsize
flags           owner    name
446cb800        1        0x60001    1        1        2048     N
B     informix rootdbs
4478dac8        2        0x68001    2        1        2048     N
SB     informix sbspace
4478dc28        3        0x42001    3        1        2048     N
TB     informix tempdbs
 3 active, 2047 maximum

Chunks
address         chunk/dbs     offset     size       free
bpages            flags pathname
446cb960        1      1      0          100000     38146
PO-B  /opt/IBM/informix/demo/server/online_root
4478dd88        2      2      0          25600      23797
23802      POSB  /opt/IBM/informix/demo/server/sbspace
                         Metadata 1745       1323       1745
4477bc30        3      3      0          5120       5067
PO-B  /opt/IBM/informix/demo/server/tempdbs
 3 active, 32766 maximum

NOTE: The values in the "size" and "free" columns for
DBspace chunks are
      displayed in terms of "pgsize" of the DBspace to which they belong.


Expanded chunk capacity mode: always
```

Note that each line displayed by the `onstat` command is larger than the page. This explains why the display is a little confusing. This command displays two different types of storage information: dbspaces and chunks.

The dbspaces of all types (dbspaces, temporary dbspaces, sbspaces (smartblob spaces)) represent a logical grouping of physical storage. So, the first section of the display lists the dbspaces. For our purpose, you only have to note the dbspace number and its name:

```
Number          Name
1               rootdbs
2               sbspace
3               tempdbs
```

A chunk is a piece of physical storage. It is either a device or a file. You may have more than one chunk in one dbspace. To find out if you are running out of space, look at the free space (in 2 KB pages in our example) and relate it to the dbspaces:

```
Chunk/dbs       size            free            pathname
1       1       100000          38146           . . .
2       2       25600           23797           . . .
3       3       5120            5067            . . .
```

In our example, we have a one-to-one relationship between dbspaces and chunks. If, for example, chunk number 1 would become full, we could use the onspaces command to add another chunk to the rootdbs dbspace and solve our lack of space problem. Let's say we want to add a 50MB chunk using the /opt/IBM/informix/demo/server/online_root2 file to the rootdbs dbspace. This can be done using:

```
$ onspaces –a rootdbs  –p
/opt/IBM/informix/demo/server/online_root2 –o 0 –s 51200
```

The command is the same if you are adding space to a sbspace.

This should be all you need to monitor and add spaces to your development environment.

Basic Understanding of Logs

When we created databases at the end of the IDS installation in Chapter 2, we made sure to turn on logging. Logging is important because it allows the database to execute multiple statements as part of a transaction and preserve database consistency in any situations.

Each time you modify data as part of information, the database server must keep track of what was modified so it can return to that version if needed. To do this, it uses log space. If the database server runs out of log space, it will hang. There are two basic things you need to do to insure you don't run out of log space. You can backup the logs and free them for reuse and you can add more logs.

Since we are talking about using Informix Dynamic Server (IDS) in development, it is not necessary to backup the log files to tape or to a directory. You could set the backup devices to NULL (/dev/null) in the `onconfig`:

- Windows: `%INFORMIXDIR%\etc\%ONCONFIG%`
- Others: `$INFORMIXDIR/etc/$ONCONFIG`

And change the parameters values `LTAPEDEV` to:

- Windows: `NULL`
- Others: `/dev/null`

The next step is to tell IDS to continuously backup the logs through the `ontape` command:

```
$ ontape -c
```

At this point, once a log is closed, it is backed up and made available for reuse. This may not be enough to avoid running out of log space. Long transactions that modify a lot of data can eat your log space and hang the server. It is possible to add logs to a running server and have the server resume operations.

There are three ways to add logs to a server. You can use the SQL API through the admin function, use the open admin tool for IDS (OAT), or use the onparams utility. Consult the appropriate documentation for the SQL API and OAT. The onparams syntax to add a log is as follows:

```
onparams –a –d <dbspace> -s <size_in_kb>
```

The command using the add option (-a), identifies a dbspace to use for the log with the –d option with a dbspace name as argument (for example `rootdbs`). It defines a size for the log file with option –s which receives a number as an argument that represents the log size in kilobytes.

Loading and Unloading Data

Loading and unloading data to and from a table is not really a database administration task, but it can actually impact the administration of the server. It is also a common operation that people may want to do in their development environment.

You can use the dbaccess load and unload commands to do this work. It is easier to look at it through an example, starting with the unload command. Let's say we want to unload the items table as a pipe separated file. We can use the following command:

```
UNLOAD TO '/tmp/file.out' DELIMITER '|'
SELECT * FROM items;
```

The default delimiter for the unload command is the pipe symbol. The command could be simplified as:

```
UNLOAD TO '/tmp/file.out' SELECT * FROM items;
```

Assuming that this command is in a file called `unload.sql`, we can execute it with the following command:

```
$ dbaccess –e stores unload
```

The command executes the `dbaccess` program asking to echo the command to the screen (-e). It used the stores database and executes the command `unload.sql`. The `.sql` extension is assumed by `dbaccess`.

The output file contains the following:

```
1|1001|1|HRO|1|250.0|
1|1002|4|HSK|1|960.0|
2|1002|3|HSK|1|240.0|
1|1003|9|ANZ|1|20.0|
2|1003|8|ANZ|1|840.0|
3|1003|5|ANZ|5|99.0|
1|1004|1|HRO|1|250.0|
2|1004|2|HRO|1|126.0|
3|1004|3|HSK|1|240.0|
. . .
```

The output file contains 67 rows. To load that file into an empty items table, we can use the command:

```
LOAD FROM '/tmp/items.out' DELIMITER '|'
INSERT INTO items;
```

Since we are using the default delimiter, we could remove that part of the statement. The `INSERT` statement could also include a list of columns if the input file did not include all the items table columns.

The command is executed through dbaccess the same way the unload command was.

We saw earlier that any changes to a table as part of a transaction require log space. If you are planning to load a large amount of data into a table, you can turn off logging under some circumstances. You can use a table type of RAW if the table does not have primary key, unique constraints, referential constraints, or indexes. You can remove these constraints before loading a table and re-enable the constraints after the load is completed. Here is an example of the commands used to load into the items table as a raw table:

```
ALTER TABLE items
DROP CONSTRAINT (u104_10, r104_11, r104_12);

ALTER TABLE items TYPE(RAW);
LOAD FROM '/tmp/items.out' INSERT INTO items;
ALTER TABLE items TYPE(STANDARD);

ALTER TABLE items ADD CONSTRAINT (
  PRIMARY KEY(item_num, order_num) CONSTRAINT u104_10,
  FOREIGN KEY(order_num)
    REFERENCES orders (order_num) CONSTRAINT r104_11,
  FOREIGN KEY(stock_num, manu_code)
  REFERENCES stock (stock_num, manu_code)
  CONSTRAINT r104_12
);
The constraints can be found in sysconstraints. Here is an SQL statement that
retrieves all the constraints on the items table:
SELECT * FROM sysconstraints
WHERE tabid = (
  SELECT tabid
  FROM systables
  WHERE tabname = 'items'
);
```

This information should make it easier to load and unload data to and from tables.

Of course, there is a lot more to know about database administration. Please consult the documentation for more information.

Reference Material

- *IBM Informix Dynamic Server Administrator's Guide, Version 11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-7748-00
- *IBM Informix Dynamic Server Administrator's Reference, Version 11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-7749-00
- *IBM Informix Guide to SQL: Reference, Version 11.50*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-7750-00
- *IBM Informix Dynamic Server DB-Access User's Guide, v11.5*
  http://www.elink.ibmlink.ibm.com/publications/servlet/pbi.wss?CTY=US&FNC=SRX&PBL=SC23-9430-00
- *Administering Informix Dynamic Server, Carlton Doe, Sept 2008,*
  ISBN 978-1-58347-076-3

IBM Global Services
Route 100
Somers, NY 10589
U.S.A.

Produced in the United States of America
June 2009

IMO14002-USEN-00

Informix Dynamic Server Application Development: Getting Started

Learn how to start developing state of the art applications using Informix Dynamic Server (IDS) as the database. It includes an introduction to IDS, its installation, the installation of the client drivers, and basic information for using IDS with Java, PHP, Ruby on Rails, and.Net.

Jacques Roy is well-known in the Informix community as an author, speaker, and active blogger. Jacques is a member of the IBM Development Competitive Technologies and Enablement Team for Informix Dynamic Server (IDS). He is the author of “IDS.2000: Server-Side Programming in C” and the lead author of “Open-Source Components for IDS 9.x”. He is also the author of multiple technical developerworks articles on a variety of subjects, and co-author of several IBM redbooks. He is a frequent speakers at data management conferences, IIUG conference and users group meetings. You can read his blog at: http://www.ibm.com/developerworks/blogs/page/jacquesroy

Informix® software

IBM