

Nextra Read Me First

Version 6
1st Edition



Contents

Chapter 1 Read Me First	2
An Open, Distributed Computing Environment.....	2
Two-Tiered (Client/Server) Architecture.....	3
Three Tiered Architecture	4
Nextra Utilities	5
Conclusion	8
Chapter 2 Installation	9
How to install	9
Chapter 3 Footnote	10
Glossary	10

Chapter 1 Read Me First

Welcome to Nextra, 3 tiered architecture based Distributed Application Server from Inspire International Inc. If you know little or nothing about this new application environment, you're reading the right book.

If you are already familiar with Nextra's development environment, this introductory book is probably not for you. Move ahead to the *Server Developer's Guide* or *Client Developer's Guide* for information that is more suited to you.

An Open, Distributed Computing Environment

Open Distributed Environment

The Nextra family of products consists of several software utilities that allow *you* to create *open, distributed, client/server* applications.

Before we continue, here is a breakdown of the highlighted terms in that last sentence.

You

"You" are the person reading this book. You are also the developer of these applications, the person who needs to know what this stuff is all about.

Distributed

"Distributed" refers to the separation of the pieces of an application. A distributed application is one whose pieces run in different logical spaces, most often on different machines. For example, you can get your data from a mainframe, process it on a workstation, and display it on a PC; the programs that run on each machine in this example are all considered part of one distributed application.

Client/Server

A client is one program, a server is another. The client may request that the server perform a specific task; as long as the server is capable, and the client has the correct permissions, the server performs the task and returns the result to the client.

Client/Server also refers to application architecture. For more information, see "Two-Tiered (Client/Server) Architecture".

On a broader scale client/server computing includes all applications that take advantage of these two types of programs—those that make requests, and those that perform them.

Open

“Open” is buzzword in the computer industry; there are open consortia, open products, open systems. (Open everything and feel a draft, right?) In general, “open” signifies a lack of lock-in to a particular vendor. Lock-in means just what it says: a dependency upon a specific combination of hardware and/or software in order for your application to work. We use “open” in the sense that our products lead you away from such vendor lock-in. Within an open distributed environment, you can connect your workstations to PCs to mainframes to Macintosh computers, often with the software you are already using. Nextra products allow you to create an open environment using the hardware and software you already have.

With these definitions in mind, take another look at the sentence being discussed.

Nextra family of products consists of several software utilities that allow *you* to create *open, distributed, client/server* applications.

Nextra is designed to help you create applications that take advantage of the latest implementations of computing technology without being locked in to one particular vendor

Two-Tiered (Client/Server) Architecture

At its inception, Client/Server computing seemed to be the answer to many of the problems facing Information Systems departments. Among other capabilities, it allowed businesses to take advantage of inexpensive desktop machines and more manageable local area networks, making downsizing a viable option for strategic computing. Software vendors soon released applications that utilized this new architecture. Typically, these applications consisted of two distinct tiers:

- a client (the user interface and most of the processing logic), and
- a server (supplier of data and some processing to the client as requested).

The client contacts the server to gain access to data; the server returns the requested information; the client then manipulates and polishes the

data for display. Multiple clients can access a single server; the server can reside on any networked machine.

As companies began to build enterprise-wide client/server applications, however, they realized that the two-tiered architecture imposed certain limitations.

Within this structure, most clients can access only their own proprietary servers. While companies have freed themselves from the hardware lock-in mandated by mainframe architecture, they have been lured into a different lock-in—one caused by software.

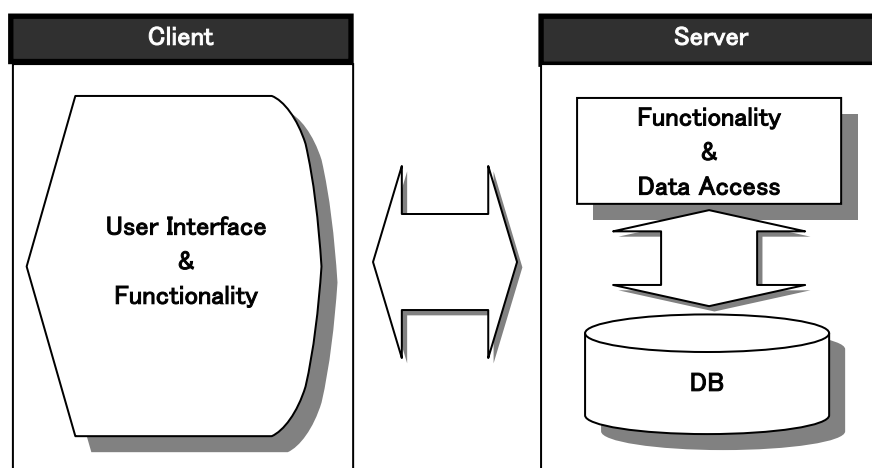


Figure 1.1: A two-tiered application

In addition to software lock-in, poor modularity limits the flexibility of applications. Code is neither reusable nor easily modified. Despite such shortcomings, some people still think that two-tiered, Client/Server applications are a viable solution for today's enterprise-wide strategic applications.

Three Tiered Architecture

Nextra has vastly improved upon the Client/Server model by isolating the processing logic from each of the two tiers into a new middle layer, called the Functionality tier. The old client becomes a pure presentation tool, while the old server becomes a raw data source, such as a Database Management System or a mainframe. The new middle tier encompasses everything the application needs to do, from retrieving data to terminal emulation, computation to transaction processing. Why is this three-tiered architecture better than the two-tiered version? Not only does a three-tiered architecture transcend two-tiered limitations, it also affords

possibilities your current client/server applications could never and will never have.

The key is the middle tier, because it allows proprietary products to interoperate. You can use a large variety of user interfaces to access different proprietary databases and remote systems of any kind. No hardware lock-in, no software lock-in. Such wide-ranging freedom of choice is unprecedented in the industry and indicates a qualitative shift in the paradigms underlying the design of information systems. Applications built with the three-tiered, Client/Server architecture are modular and easy to change. Additional clients and servers, written in almost any language or presentation script, can be added or subtracted to your distributed application *while it is running*. And there is no need to modify the other parts of the application when you fit in a new component.

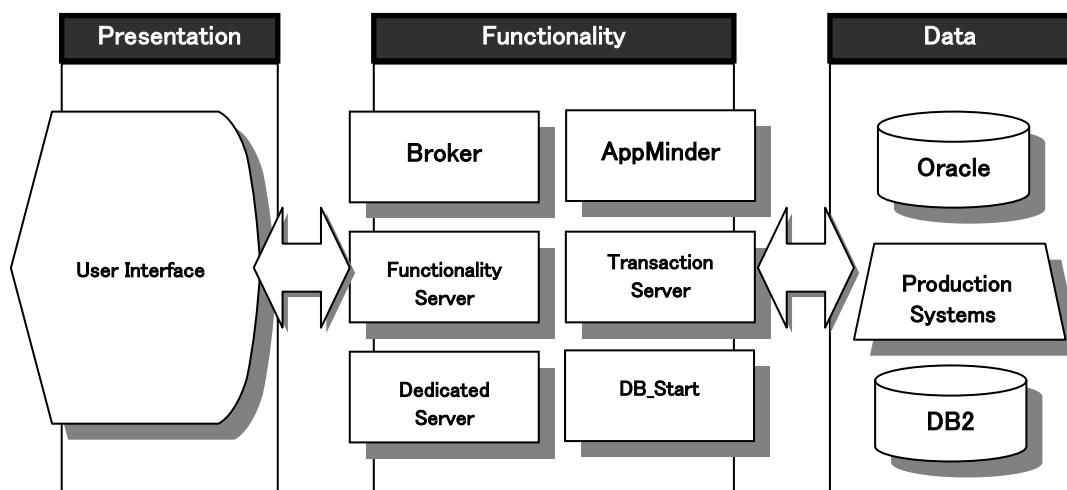


Figure 1.2: A three-tiered application

The three-tiered, Client/Server model forms the foundation of Nextra's open distributed environment. Nextra applications are secure, robust, and provide excellent performance.

Nextra utilities allow you to build and modify applications easily, integrating legacy platforms when necessary and supporting new technologies as they emerge.

Nextra Utilities

The open distributed environment is a development environment that makes it easier to create distributed applications which run over your

existing hardware, software, and network. Here is a preview of some of the utilities in your package:

RPCMake

In order for a distributed application to run, the pieces need to communicate over the lines of a network. Coding network communication routines can be a hassle. So we decided to do it for you.

The **RPCMake** utility generates auxiliary pieces of code called client stub and server skeleton when it is given a server definition file called IDL (Interface Definition Language) file. When the client stub is included in the client program and the business logic is built on top of the server skeleton, the two parties can talk to each other: Yes, that simple.

SQLMake

The **SQLMake** utility aids in building servers to access SQL databases. It generates an IDL(Interface Definition Language) file of modified ANSI SQL queries and optionally generates client application stubs.

TPMake

The **TPMake** utility takes **SQLMake** one step further – it generates C code for creating DB servers that have transaction processing capabilities, server code a header file and a Makefile.

For **TPMake**, you need to create a resource map file in addition to the file of SQL statements. The resource file is a text file that maps each SQL statement file to the correct DB.

RPCDebug

The **RPCDebug** is a testing utility with a GUI. With it, you can point-n-click your way through the various RPCs of a particular server. It allows you to test the reliability of a server's function calls in a controlled distributed environment, before you deploy your application.

RPC Developer

The **RPC Developer** is an integrated development environment for all utilities: **RPCMake**, **SQLMake**, **TPMake** and **RPCDebug**.

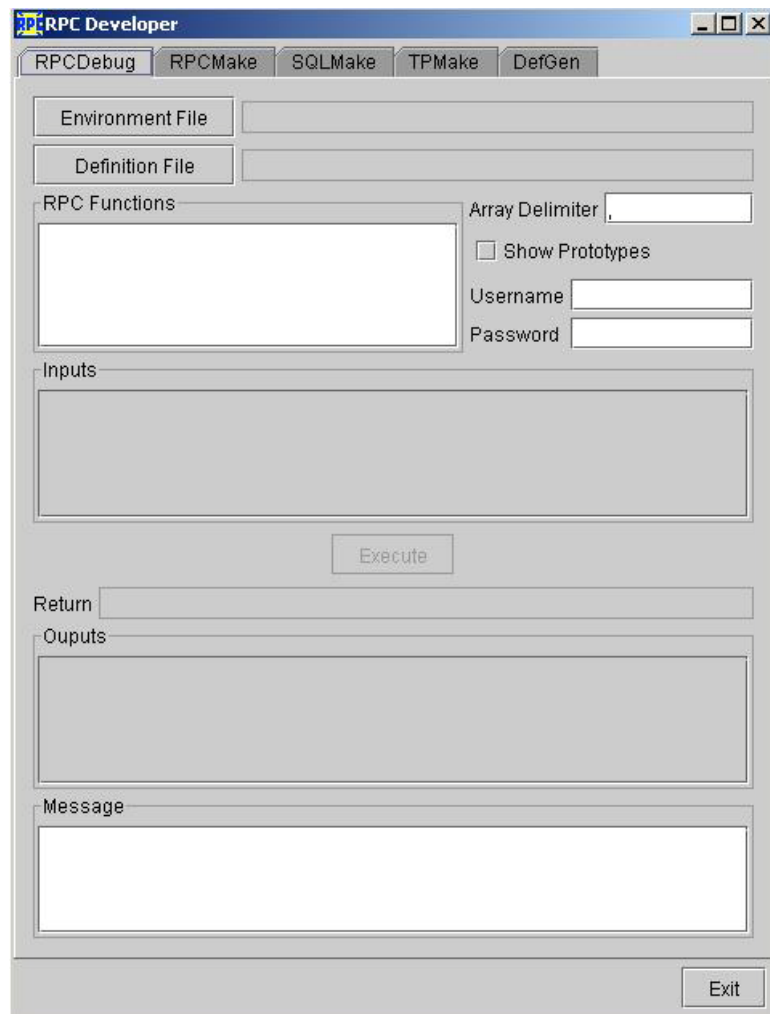


Figure 1.3: RPC Developer GUI

Broker

The **Broker** is sometimes referred to as a “naming service” which caches the services’ location information in its memory. Services always register the location information to the **Broker** upon start-up. The Client in the Network requests the **Broker** first the services’ location information before accessing the services. It is possible to create hierarchies of **Brokers**, with a master **Broker**, parent **Brokers** and Sub-**Brokers**.

DB_start

The ***DB_starts*** act as start-up utilities specifically for database servers. There is a start utility for each of the databases supported by the Nextra Developer Package: Oracle, DB2, SQL Server and HiRDB.

AppMinder

AppMinder provides network management services for distributed applications within our open distributed environment. It has the ability to start Brokers and servers for you, and to monitor these services once they are up.

Conclusion

The next step for the beginner is to read the *Server Developer's Guide*.

Chapter 2 Installation

How to install

Please refer to Install.txt in the product CD.

Chapter 3 Footnote

Glossary

This chapter contains definitions of terms used throughout the Nextra documentation set.

Table 3.1: Definitions

Word	Definition
Application	Each distributed application consists of some combination of clients and servers, communicating through well-defined remote interfaces.
Cell	Cell is a unit comprising Broker(s) and the servers registered to it. There must be one Master Broker at least in the Cell and Sub-Broker and servers registered to the Master Broker.
Clustering	Or called "Application Clustering". Having equal or more than 2 Cells, we call "Clustering".
Client	A component of a distributed application that makes requests on behalf of a user. Each client may utilize more than one service.
DB Access Server	See DB_start .
DB Access Modules	Utilities which lets you access RDBMSs Nextra provides 2 utilities: DB Access Server and Transaction Server.
Dedicated Serve	This server creates and assigns one child process exclusively for each client.
Functionality Server	Naming for generic server sit in the mid-tier / Functionality tier in the 3 tiered architecture. Functionality server is basically not accessing to entities in DB tier.
IDL	Interface Definition Language
RPC	Remote Procedure Call. A function call that invokes a procedure in a different program, exchanging data without regard to operating system or hardware differences. Calling syntax is the same as for a local function call.

Server	A component of a distributed application that responds to client requests, and provides access to shared resources. A server is an instance of a service.
Service	A collection of one or more interfaces that implement some logically related set of functions in a distributed application. Each server provides one service.
Skeleton	The portion of a server program that executes the data marshalling and network transportation routines. Always generated by Nextra/RPCMake utility.
Stub	The portion of a client program that executes the data marshalling and network transportation routines. Always generated by Nextra/RPCMake utility.
Transaction Server	<p>A software component that is used in implementing transactions. A transaction involves multiple parts which must be completed atomically, for example when paying someone from your bank the system must guarantee that the money is taken from your account and paid into the other persons account. It would simply be unacceptable for just one or the other action to take place.</p> <p>This will mean ensuring that transactions are guaranteed, or that if a transaction fails the system can tell this has happened.</p> <p>In the case of a transaction failing it can then be "backed out", which will mean that the system reverses all the actions that happened during the partial completion of the transaction.</p> <p>This is sometimes referred to as the ACID property.</p>
Variable Named Server	Server created with an IDL file containing a variable for the interface name. Thus even from the single executable, you can start up server processes in different server/interface names.

Nextra **Read Me First**

2011.09.15	v6 1 st Edition
2008.10.15	v5 2 nd Edition
2007.03.30	3 rd Edition
2006.11.21	Additional information as to environment set-up after Installation
2004.09.07	Modification to Installer
2004.07.19	2 nd Edition
2003.04.08	1 st Edition

Author: Inspire International Inc.

Copyright © 1998-2011 Inspire International Inc.
Printed in Japan