



Installation and Platform Notes for UNIX

Release 10.2

Installation and Platform Notes for UNIX

Part Number: 10.2-IUNIX-0

Release 10.2, October 13, 2011

The information in this document is subject to change without notice. Objectivity, Inc. assumes no responsibility for any errors that may appear in this document.

Copyright 1993–2011 by Objectivity, Inc. All rights reserved. This document may not be copied, photocopied, reproduced, translated, or converted to any electronic or machine-readable form in whole or in part without prior written approval of Objectivity, Inc.

Objectivity and Objectivity/DB are registered trademarks of Objectivity, Inc. Active Schema, Objectivity/DB Active Schema, Assist, Objectivity/Assist, ooAssistant, Objectivity/DB ooAssistant, Objectivity/DB Fault Tolerant Option, Objectivity/FTO, Objectivity/DB Data Replication Option, Objectivity/DRO, Objectivity/DB High Availability, Objectivity/HA, Objectivity/DB Hot Failover, Objectivity/DB In-Process Lock Server, Objectivity/IPLS, Objectivity/DB Open File System, Objectivity/OFS, Objectivity/DB Parallel Query Engine, Objectivity/PQE, Objectivity/DB Persistence Designer, Objectivity/DB Secure Framework, Objectivity/Secure, Objectivity/C++, Objectivity/C++ Data Definition Language, Objectivity/DDL, Objectivity/Dashboard, Objectivity/C++ Active Schema, Objectivity/C++ Standard Template Library, Objectivity/C++ STL, Objectivity/C++ Spatial Index Framework, Objectivity/Spatial, Objectivity for Java, Objectivity/.NET, Objectivity/.NET for C#, Objectivity/Python, Objectivity/Smalltalk, Objectivity/SQL++, Objectivity/SQL++ ODBC Driver, Objectivity/ODBC, Objectivity Event Notification Services, and Persistence Designer are trademarks of Objectivity, Inc.

Other trademarks and products are the property of their respective owners.

ODMG information in this document is based in whole or in part on material from *The Object Database Standard: ODMG 2.0*, edited by R.G.G. Cattell, and is reprinted with permission of Morgan Kaufmann Publishers. Copyright 1997 by Morgan Kaufmann Publishers.

The software and information contained herein are proprietary to, and comprise valuable trade secrets of, Objectivity, Inc., which intends to preserve as trade secrets such software and information. This software is furnished pursuant to a written license agreement and may be used, copied, transmitted, and stored only in accordance with the terms of such license and with the inclusion of the above copyright notice. This software and information or any other copies thereof may not be provided or otherwise made available to any other person.

RESTRICTED RIGHTS NOTICE: Use, duplication or disclosure by the U.S. Government is subject to restrictions set forth in the Objectivity, Inc. license agreement and as provided in DFARS 227.7202-1(a) and 227.7202-3(a) (1998), and FAR 12.212, as applicable. Objectivity, Inc., 640 West California Avenue, Suite 210, Sunnyvale, CA 94086-3624.

Contents

About This Book	9
Audience	9
Organization	9
Conventions and Abbreviations	10
Getting Help	11
Chapter 1 Objectivity/DB Installation	13
System Requirements	14
Accessing Objectivity Installation Files	15
Installing Objectivity/DB	16
Preparing to Use Objectivity/DB	18
Setting Up a License File	21
Starting Objectivity/Assist	22
Setting Up the Lock Server	24
Setting Up Data-Server Software	26
Setting Up ootoolmgr	29
Troubleshooting Installation	31
Chapter 2 Objectivity/C++ Installation	35
System Requirements	35
Installing Objectivity/C++ or Objectivity/DDL	36
Preparing to Use Objectivity/C++ or Objectivity/DDL	37
Testing Objectivity/C++ Setup	38

Chapter 3	Objectivity/DB Active Schema for C++ Installation	41
	System Requirements	41
	Installing Active Schema for C++	41
	Preparing to Use Active Schema for C++	42
Chapter 4	Objectivity for Java Installation	45
	System Requirements	45
	Installing Objectivity for Java	46
	Preparing to Use Objectivity for Java	47
	Testing Objectivity for Java Setup	48
Chapter 5	Objectivity/DB Active Schema for Java Installation	51
	System Requirements	51
	Installing Active Schema for Java	52
	Preparing to Use Active Schema for Java	52
Chapter 6	Objectivity/Smalltalk for VisualWorks Installation	53
	System Requirements	53
	Installing Objectivity/Smalltalk for VisualWorks	54
	Preparing to Use Objectivity/Smalltalk for VisualWorks	54
	Setting Up VisualWorks	55
	Setting Up VisualWorks With ENVY/Developer	56
	Testing Objectivity/Smalltalk for VisualWorks Setup	56
Chapter 7	Objectivity/Python Installation	59
	System Requirements	59
	Installing Objectivity/Python	60
	Preparing to Use Objectivity/Python	61
	Setting Up Python	62

Chapter 8	Objectivity/SQL++ Installation	67
	System Requirements	67
	Installing Objectivity/SQL++	68
	Preparing to Use Objectivity/SQL++	69
	Setting Up the Objectivity/SQL++ ODBC Server	70
	Testing Interactive SQL++	73
	Preparing the ODBC Server for Testing	74
Chapter 9	Objectivity/SQL++ ODBC Driver Installation	77
	System Requirements	77
	Installing Objectivity/ODBC	78
	Preparing to Use Objectivity/ODBC	79
	Checking and Registering the Driver	80
	Adding Objectivity/DB Data Sources	81
	Configuring TCP/IP	83
	Testing Objectivity/ODBC	84
Chapter 10	Objectivity/DB High Availability Installation	87
	System Requirements	87
	Installing Objectivity/HA	88
	Preparing to Use Objectivity/HA	89
Chapter 11	Objectivity/DB Parallel Query Engine Installation	91
	System Requirements	91
	Installing Objectivity/PQE	91
	Preparing to Use Objectivity/PQE	92
	Setting Up the Query Server	93
	Using Objectivity/PQE	94
Chapter 12	Objectivity/IPLS Installation	95
	System Requirements	95
	Installing Objectivity/IPLS	95
	Preparing to Use Objectivity/IPLS	96
	Using Objectivity/IPLS	97

Appendix A C++ Application Development 99

Linking Applications to Objectivity/DB	99
Libraries for Static Linking	100
Linking to Shared Libraries	101
Link Rules for End-User Environments	103
Linking With Purify	103
Linking to Additional Objectivity Products and Features	104
Linking a Database Application	104
Linking a Lock-Server Performance-Monitoring Program	105
Using Makefiles	106
Compiler Flags	106
Application Programming Issues	107
Signal Handling	107
Stack Size for Multithreaded Applications	107
File Descriptor Limit	107
Debugging an Application	108
Preparing to Debug an Application	108
Printing Handles	108
Printing Objects	109

Appendix B Java Application Development 111

Application Programming Issues	111
Signal Handling	111
File-Descriptor Limit	112
Finalizing Objects in JDK 1.6	113
Java Command Line Options	114
Stack Size	114
Memory Requirements	114
HP-UX Architectures	115
64-bit Architectures	115

Appendix C Sample Applications for C++ and Java	117
Overview of Sample Applications	118
Sample Federated Databases	118
Appendix D Troubleshooting an Application	121
Index	123

About This Book

This book, *Installation and Platform Notes for UNIX*, describes how to install Objectivity products on supported UNIX platforms. This book also provides platform-specific information that supplements the information in the rest of the document set.

Audience

This book is intended for administrators or developers who install Objectivity products. This book assumes you are familiar with the operating system and any specified prerequisite software (such as TCP/IP) on the installation platforms.

The appendixes of this book are intended for developers who create Objectivity/DB database applications on UNIX platforms. The appendixes assume you are familiar with your compiler and development environment.

Organization

Each of the numbered chapters describes the requirements and steps for installing a particular Objectivity product on UNIX platforms.

Appendix A provides platform-specific details that supplement the information in the Objectivity/C++ books. Topics include compilation and link rules, C++ programming issues, and building and debugging C++ database applications.

Appendix B provides platform-specific details that supplement the information in the Objectivity for Java programmer's guide.

Appendix C describes the sample C++ and Java database applications.

Appendix D provides troubleshooting guidelines.

Conventions and Abbreviations

Navigation

In the online version of this book, table of contents entries, index entries, cross-references, and underlined text are hypertext links.

Typographical Conventions

<code>oonewdb</code>	Command, literal parameter, code sample, filename, pathname, output on your screen, or Objectivity-defined identifier
<code>installDir</code>	Variable element (such as a filename or a parameter) for which you must substitute a value
Browse FD	Graphical user-interface label for a menu item or button
<i>lock server</i>	New term, book title, or emphasized word

Abbreviations

<i>(administration)</i>	Feature intended for database administration tasks
<i>(HA)</i>	Feature of the Objectivity/DB High Availability product
<i>(IPLS)</i>	Feature of the Objectivity/DB In-Process Lock Server product
<i>(ODMG)</i>	Feature conforming to the Object Database Management Group interface
<i>(PQE)</i>	Feature of the Objectivity/DB Parallel Query Engine product

Command Syntax Symbols

[...]	Optional item. You may either enter or omit the enclosed item.
{...}	Item that can be repeated.
... ...	Alternative items. You should enter only one of the items separated by this symbol.
(...)	Logical group of items. The parentheses themselves are not part of the command syntax; do not type them.

Command and Code Conventions

In code examples or commands, the continuation of a long line is indented. Omitted code is indicated with the ellipsis (...) symbol. “Enter” refers to the standard key (labeled either Enter or Return) for terminating a line of input.

Getting Help

We have done our best to make sure all the information you need to install and operate Objectivity products is provided in the product documentation. However, we also realize problems requiring special attention sometimes occur.

Technical Support Web Site

You can find answers to frequently asked questions, supported platforms, known bugs, and bug fixes on the Objectivity Technical Support Web site. Send electronic mail or call Objectivity Customer Support to gain access to the site.

How to Reach Objectivity Customer Support

You can contact Objectivity Customer Support by:

- **Telephone:** Call 1.408.992.7100 or 1.800.SOS.OBJY (1.800.767.6259) Monday through Friday between 6:00 A.M. and 6:00 P.M. Pacific Time, and ask for Customer Support.
The toll-free 800 number can be dialed *only* within the 48 contiguous states of the United States and Canada.
- **Fax:** Send a fax to Objectivity at 1.408.992.7171.
- **Electronic Mail:** Send electronic mail to help@objectivity.com.

Before You Call

Please be ready to submit the following to Objectivity Customer Support:

- Your name, company name, address, telephone number, fax number, and email address
- Detailed description of the problem you have encountered
- Information about your workstation environment, including the type of workstation, its operating system version, and compiler or interpreter
- Information about your Objectivity products, including the version of the Objectivity/DB libraries

You can use the Objectivity/DB `oosupportinfo` tool to obtain information about your workstation environment and your Objectivity products.

Developer Network Web Site

The Objectivity Developer Network Web site provides technical information and resources, such as tutorials, FAQs, code examples, and sample applications, for Objectivity/DB developers and integrators.

<http://devnet.objectivity.com>

Objectivity/DB Installation

This chapter describes the requirements and steps for installing Objectivity/DB on a UNIX platform. Objectivity/DB is an object database management system that enables your applications to create and access persistent objects in a *federated database*. As the foundation of the Objectivity product set, Objectivity/DB provides:

- Tools for database administration and data inspection.
- Servers for managing concurrency and accessing remote files.
- Runtime libraries containing the Objectivity/DB *kernel*, which is used by the tools and servers, and by the database applications you develop.
- Programming interface for custom programs that monitor how database applications use the servers that manage concurrency. The information collected by such programs can help you analyze application performance.

System Requirements

You can install Objectivity/DB on the UNIX architectures listed in Table 1-1. Each architecture represents a combination of hardware, UNIX operating system, and addressing mode (32-bit or 64-bit).

Table 1-1: Supported UNIX Architectures

Architecture Name	Hardware
hprisc	HP 9000 Series 700/800
hpuxia64	Intel Itanium 2
linux86gcc3	x86 (586 or greater)
linux86_64	x86_64
solaris7	Sun SPARC
solaris86_64	x86_64
sparc64	Sun SPARC

See the Objectivity Technical Support Web site for the currently supported versions of the operating system for each architecture. Contact Objectivity Customer Support to get access to this Web site.

Software Requirements

Objectivity/DB requires that the following software be installed on your system:

- C++ shared libraries (required for the Objectivity/DB kernel and tools)
- X Window System (required for running Objectivity/DB graphical tools)
- Java runtime environment (required for running Objectivity/Assist, a graphical interface for browsing and editing for federated databases)

Accessing Objectivity Installation Files

You can access Objectivity installation files (including the Objectivity installation script) as described in the following subsections.

Downloading From the Objectivity Technical Support Web Site

1. Access the download page on the Objectivity Technical Support Web site:
`http://support.objy.com/new/download`
2. Click the link corresponding to your UNIX platform to download the compressed Objectivity distribution file to the computer running your Web browser.
3. If necessary, move the distribution file to a directory on your UNIX workstation.
4. Uncompress the distribution file and extract the Objectivity installation files from it. For example, enter:

```
gunzip -c filename | tar -xvf -
```

Mounting an Objectivity Distribution CD

1. Place the Objectivity distribution CD in the CD-ROM drive.
2. If necessary, enter a command to explicitly mount the CD.
See your operating system documentation for the proper mount command. On the `hprisc` and `hpuxia64` architectures, use the `-o cdcase` mount option to suppress version numbers and case changes in filenames.

Installing Objectivity/DB

You can install Objectivity/DB either alone or in combination with one or more other products.

Before You Install

1. Verify that your workstation meets the system requirements for Objectivity/DB (see “System Requirements” on page 14).
2. If you are installing multiple Objectivity products at this time, verify that the system requirements for those products are met (see the installation chapter for each such product in this book).
3. If you have not done so already, obtain the Objectivity installation files as described in “Accessing Objectivity Installation Files” on page 15.
4. Locate or create an installation directory *installDir* for Objectivity/DB—for example, */usr/object*.

The installation script will automatically create and use a subdirectory called *installDir/arch*, where *arch* is the architecture name for your platform shown in Table 1-1.

NOTE To preserve an existing Objectivity/DB installation, you can create a new installation directory or rename the architecture-specific subdirectory in the existing *installDir*—for example, by moving *installDir/arch* to *installDir/arch.old*.

Running the Objectivity Installation Script

1. Log in to your workstation as `root`.
2. Change to the directory containing the uncompressed installation files. If you mounted a CD, change to the CD mount directory.
3. Run the Objectivity installation script with a command such as the following:
`./install.sh`
4. When prompted, select one of the following installation options by entering its item number:
 - Select **Simple Installation**, then select a programming language (C++, Java, Python, or Smalltalk), to install just the essential Objectivity products for developing applications in the selected language.
 - Select **Full Installation** to install all Objectivity products.
 - Select **Custom Installation** to display a list of installable Objectivity products. Enter the item number for Objectivity/DB and any other products you want to install at this time. Separate multiple item numbers by commas.

NOTE You must be licensed for every Objectivity product you install.

5. At the directory prompt, specify the chosen installation directory `installDir`. If you get an error message reporting insufficient disk space or incorrect permissions, you can specify another directory.
6. (Optional) Specify include paths and preprocessor variables for the Objectivity/DDL preprocessor or press return to accept the current values.

The installation script:

- Creates the subdirectory `installDir/arch` and copies the product release files into it.
- Runs the installation verification test `ooverify`. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 31.

Now perform the steps in “Preparing to Use Objectivity/DB” below.

Preparing to Use Objectivity/DB

After the installation script completes, perform the following steps:

1. Familiarize yourself with your installation by reading “Release Files for Objectivity/DB” on page 20.
2. Set up your Objectivity license file, if you have not already done so; see “Setting Up a License File” on page 21.
3. Give all Objectivity/DB users both read and execute permission to *installDir*.
4. Advise each application developer to:
 - Add *installDir/arch/bin* to the PATH environment variable.
 - Add *installDir/arch/lib* as the first component of the environment variable indicated below (required for running Objectivity/DB tools).

LD_LIBRARY_PATH On linux86_64, linux86gcc3, solaris7,
 sparc64, solaris86_64

SHLIB_PATH On hprisc and hpuxia64

LD_LIBRARY_PATH_64 On sparc64 or solaris86_64.

Note: LD_LIBRARY_PATH_64 must be set for only the Solaris 10 operating system.

5. (*Selected architectures only*) Verify that you can start Objectivity/Assist, and perform any necessary setup; see “Starting Objectivity/Assist” on page 22.
Note: If you are not sure whether Objectivity/Assist is available on your architecture, check for the directory *installDir/arch/assist*. If the directory does not exist, Objectivity/Assist is not available; go on to step 4.
6. Set up the lock server to manage concurrent database access; see “Setting Up the Lock Server” on page 24.
7. Set up data-server software (NFS or AMS) to provide remote data access; see “Setting Up Data-Server Software” on page 26.
8. Set up the *ootoolmgr* tool for Objectivity/DB administration and data browsing; see “Setting Up ootoolmgr” on page 29.
Note: If Objectivity/Assist is available on your architecture, you can use it instead of *ootoolmgr* and skip this step.
9. If you installed multiple Objectivity products, read the installation chapter for each product, and follow the steps in the chapter’s “Preparing to Use” section.

10. Familiarize yourself with Objectivity online books. To do so, start Acrobat Reader and display the file *installDir/doc/ObjyBooks.pdf*. A PDF file is displayed containing links to the online books.

NOTE All Objectivity online PDF books are installed with Objectivity/DB. You can delete the files for books you don't want. The book for an individual product is reinstalled if you subsequently install that product.

11. Read:

- *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the file *installDir/arch/doc/ReleaseNotes.pdf*.
- The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

12. (Optional) If Objectivity/Assist is available on your architecture, use it to perform the Objectivity/DB Basics Tutorial for getting started with Objectivity/DB concepts.

NOTE If you have existing data, tools, or applications from an earlier release of Objectivity/DB, and you plan to use them with the current release, read Chapter 3, "Release Compatibility," of *Objectivity Release Notes*. You may need to perform an upgrade or be aware of limitations.

Release Files for Objectivity/DB

Objectivity/DB executables, libraries, and online books are organized in subdirectories of the *installDir/arch* directory as shown in Table 1-2. This directory structure is the same for all architectures, which means you can:

- Install versions of Objectivity/DB for different architectures in a shared file system.
- Develop applications that will compile on other platforms without having to consider workstation-specific directory naming conventions.

Table 1-2: Objectivity/DB Release Files in *installDir/arch*

Subdirectory	Contains
bin	Executables for Objectivity/DB tools and servers
doc	PDF files for the online books of Objectivity products PDF file for <i>Objectivity Books</i> , which has links to the online books Subdirectories for HTML documentation sets
assist (Selected architectures)	Subdirectories for Objectivity/Assist plug-ins
	Executable for Objectivity/Assist
etc	Files and programs supporting various Objectivity tools and products
include	Include file for the lock-server performance-monitoring interface
lib	Shared libraries for Objectivity/DB tools and the Objectivity/DB kernel; libraries for object qualification and Objectivity/DB performance analyzer
	Static and shared libraries for linking custom lock-server performance-monitoring programs
plugins	Subdirectory for plugin specification files
samples	Sample database applications for demonstration and testing

Setting Up a License File

After the installation script completes, you should set up a license file containing your encrypted Objectivity license. Setting up a license file makes your license available for initializing new Objectivity/DB federated databases or upgrading existing ones. Every federated database must contain an Objectivity license that authorizes access by Objectivity/DB tools and applications. For more information about federated-database licensing, see Chapter 4, “Federated Database Tasks,” in *Objectivity/DB Administration*.

You normally receive an encrypted version of your current Objectivity license whenever you acquire one or more new Objectivity products, or whenever you upgrade to new major releases of the products you already have. In most cases, you receive the encrypted license by electronic mail, although other arrangements may be made.

To set up a license file for a new Objectivity license:

1. Create a *default license file*:
 - a. Check whether a file called `oolicense.txt` already exists in your Objectivity/DB installation directory `installDir`. If so, move or rename that file.
 - b. Save your new Objectivity license as a text file called `oolicense.txt` in your Objectivity/DB installation directory `installDir`.
Note: If you received your encrypted license by electronic mail, you can simply save the attached file from your mail reader, or you can create an empty text file and paste the license text into it.
2. Make a backup copy of the new license file in another directory for safekeeping. A useful convention is to include the license’s identifier in the filename. (The identifier is included in the electronic mail message or other file accompanying the encrypted license.)

If your site conventions require it, you can choose a nondefault name and location for your license file; you must then specify the license file explicitly whenever you create or upgrade a federated database.

NOTE If you are authorized to use multiple Objectivity products, all authorization is combined in a single license. Thus, you need to set up only one license file for your current combination of products (rather than setting up a separate license file for each product).

Starting Objectivity/Assist

(*Selected architectures only*) Objectivity/Assist (Assist) is a graphical tool for getting started quickly with Objectivity/DB. You can use Assist to perform tasks such as:

- Creating and administering a federated database.
- Listing a federated database's physical components (files and servers) and logical components (databases, containers, and persistent objects).
- Creating, browsing, and changing the data in a federated database, and the types in a federated database's schema.
- Generating C++ and Java class definitions (including accessor methods) from a federated database's schema.

Assist also provides tutorials for getting started with Objectivity/DB basic concepts, Objectivity/C++ application development, and Objectivity for Java application development.

Assist is implemented as a *rich client application* on the [Eclipse Rich Client Platform \(RCP\)](#).

Starting Objectivity/Assist

To start Assist:

1. At a command prompt, enter:
`installDir/arch/assist/ooassist`
2. At the prompt, specify a *workspace directory* for Assist project files (see below).

Assist Workspace Directory

Any directory on your computer can serve as your Assist workspace directory. For example, to keep your Assist files with your Objectivity/DB installation, you can specify `installDir/arch/assist/workspace`. If necessary, a directory with the name you specified is created.

NOTE Every user or group that is to run Assist must have read and write permissions on the hidden directory `.metadata` within the Assist workspace directory.

Simplifying Assist Startup

You can simplify Assist startup as follows:

1. Add `installDir/arch/assist` to your PATH environment variable.
2. At a command prompt, enter:

```
ooassist
```

Configuring Your Own Eclipse Platform

If your computer already has a compatible release of the Eclipse Platform or Eclipse SDK, you can install the Assist plugin into that framework:

1. Copy all the subdirectories in `installDir\arch\assist\plugins` that begin with `com.objy.assist` *except* `com.objy.assist.rcp_x.x.x` into the `eclipseInstallDir\plugins` directory, where:

`installDir/arch` Objectivity/DB installation directory and subdirectory for your UNIX architecture

`eclipseInstallDir` Directory containing the Eclipse installation to be used

2. Copy the `installDir\arch\assist\features\com.objy.assist_x.x.x` directory into the `eclipseInstallDir\features` directory.
3. Start Eclipse.
4. From the Eclipse menus, choose **Window > Open Perspective > Other**, then choose **Objectivity/Assist**.

Getting Started With Assist's Tutorials

Assist provides several tutorials for getting started with Objectivity/DB. To find a tutorial, click on its link in Assist's **Welcome** page. Alternatively, you can:

1. Start Assist, if you have not already done so.
2. In the Assist menu bar, click **Help > Help Contents**.
3. In the Help window, click **Objectivity/Assist User Guide**. If you are new to Objectivity/DB, start with the **Objectivity/DB Basics Tutorial**.

Setting Up the Lock Server

Objectivity/DB uses a system process called a *lock server* to manage concurrent access to persistent objects in one or more federated databases. For information about lock servers, see Chapter 8, “Using a Lock Server,” of *Objectivity/DB Administration*.

You can run the lock server on one or more hosts. If you want to run a lock server locally on a workstation that does not contain the Objectivity/DB installation, you can copy the `oolockserver`, `ools`, and `oolsrec` executables to that workstation.

For security purposes, you should consider running the lock server under a special-purpose user account in a group that can be granted just the minimum necessary permissions. These permissions are described in Chapter 8, “Using a Lock Server,” of *Objectivity/DB Administration*.

You may start a lock server at any time after installation, but you must start it before running any database applications that require concurrency management. It is common practice to configure your workstation so that the lock server starts whenever the machine reboots.

Perform the following steps when you start the lock server on a particular workstation for the first time:

1. Set up a log directory for lock-server log files. Follow the steps in “Setting Up a Log Directory for the Lock Server” on page 25.
2. Start the lock server to verify that it is installed properly. To start the lock server under the user account `userName`, you can enter a command such as the following:

```
su - userName -c "installDir/arch/bin/oolockserver"
```

3. If the lock server does not start, verify that you performed steps 1 and 2 correctly, and try starting the lock server again.

The lock server will not start if another service already uses the TCP/IP port that the lock server expected to use. If you cannot reassign the other service to a different port, you may have to change the default port for the lock server. To do this, see Chapter 8, “Using a Lock Server,” in *Objectivity/DB Administration*.

4. (Optional) Configure your workstation to start the lock server whenever the machine reboots. To do this, edit the workstation’s startup script (usually `/etc/rc.local` or `/etc/init.d`) and add a command such as the one in step 2.

See Chapter 13, “Automatic and Manual Recovery,” in *Objectivity/DB Administration* for information about entering a command to start the lock server upon reboot with automatic recovery enabled.

Setting Up a Log Directory for the Lock Server

A running lock server creates log files in a *log directory*. By default, the lock server expects to write log files in the default log directory, which is `/usr/spool/objy`. Alternatively, you can arrange for the lock server to write these files into a nondefault log directory.

NOTE If you run the lock server and AMS on the same host, they will both use the same log directory for their log files.

Perform the following steps to set up the log directory *logDir*, where *logDir* could be either the default log directory `/usr/spool/objy` or a nondefault log directory:

1. Log in as `root`.
2. Verify that the desired log directory exists, and, if necessary, create it. Enter:

```
mkdir logDir
```
3. Set the log directory's permissions to enable the lock server to create and update files in this directory, and to enable all users to read and write files there. Enter:

```
chmod 777 logDir
```

If you set up a nondefault *logDir*, you must enable the lock server to find it. To do this:

- ▶ Set the environment variable `OO_SERVER_LOG_DIR` to *logDir* before you start the lock server.

Additionally, if you are configuring your workstation to restart the lock server upon reboot, you must arrange for the environment variable to be set before the lock server is restarted automatically. You can do this by editing a startup file (such as `.login`) for the *userName* account, and adding a command to set the environment variable.

NOTE The lock server will consult the `OO_SERVER_LOG_DIR` environment variable before it looks for the default log directory.

For information on changing the log directory for lock-server log files, see Chapter 8, "Using a Lock Server," in *Objectivity/DB Administration*.

Setting Up Data-Server Software

You can distribute an Objectivity/DB system across multiple hosts in a network environment. This means that an Objectivity/DB application running on one host (called the *client host*) can access Objectivity/DB files on other hosts (called *data server hosts*). These files include data files and journal files.

By default, Objectivity/DB applications contact NFS on a remote data-server host to access the Objectivity/DB files there. For improved update performance, you can run the Advanced Multithreaded Server (AMS) on hosts containing Objectivity/DB files. When AMS is running, it is contacted instead of NFS for remote data access. Within a distributed Objectivity/DB system, you can use NFS on some hosts and AMS on others. Note, however, that you *must* run AMS on every host that is to store an autonomous partition or a replicated database; see Chapter 10, “Objectivity/DB High Availability Installation” in this book. For more information about AMS, see *Objectivity/DB Administration*.

Setting Up NFS

Perform the following steps on each UNIX data-server host that is to provide file access to remote Objectivity/DB applications through NFS:

1. Find out whether the `nfs` and `mountd` daemons are running. Enter:
`rpcinfo -p hostname`
2. If necessary, start the `nfs` and `mountd` daemons. (The `mountd` daemon may require an option on some architectures; see “Non-Privileged Ports” below.)
3. Verify that NFS exports any directories that will contain Objectivity/DB data files or journal files. For example, list these directories in the `/etc/exports` file. (On some architectures, you may need to include an additional option; see “Non-Privileged Ports” below.)

Non-Privileged Ports

Depending on the architecture of your UNIX data-server host, you may need to take an extra step to ensure that all remote Objectivity/DB applications can access the NFS-exported directories. In particular, you need to arrange for NFS

on such data-server hosts to accept client requests originating from non-privileged ports, as described in Table 1-3.

Table 1-3: Additional NFS Setup on Selected Data-Server Architectures

Architecture of Data-Server Host	Allow Client Requests from Non-Privileged Ports
hprisc	Run <code>mountd</code> with the <code>-p</code> option.
hpuxia64	Run <code>mountd</code> with the <code>-p</code> option.
linux86_64	Export Objectivity/DB directories using the <code>insecure</code> option.
linux86gcc3	Export Objectivity/DB directories using the <code>insecure</code> option.

Setting Up Client Hosts for Remote Data Access Through NFS

When you use NFS to provide data access in a distributed Objectivity/DB system, you may need to adjust the data packet size on each client host that will access files through NFS. In particular, you should check whether the TCP/IP protocol stack on each client host requires a smaller data packet size than 8192 bytes, which is the default used by Objectivity/DB with NFS. Furthermore, in a congested network, a Remote Procedure Call (RPC) timeout error message may also indicate that the data packet size is too large.

To adjust the data packet size:

- Set the environment variable `OO_NFS_MAX_DATA`.

NOTE You do not need to adjust the data packet size if you use AMS instead of NFS.

Setting Up the Advanced Multithreaded Server

Perform the steps given below on each UNIX host that is to provide file access to remote database applications through AMS. If you want to run AMS locally on a workstation that does not contain the Objectivity/DB installation, you can copy the `oostartams` and `ooams` executables to that workstation, along with the shared libraries they reference (use a command such as `ldd` to identify the required libraries).

For security purposes, you should consider running AMS under a special-purpose user account in a group that can be granted just the minimum necessary permissions. These permissions are described in Chapter 9, “Advanced Multithreaded Server,” in *Objectivity/DB Administration*. When an application

uses AMS, any data files or journal files created by the application are owned by the user account under which AMS was started.

NOTE It is strongly recommended that you do *not* run AMS under the `root` account.

You may start AMS at any time after installation, but before running any client applications. It is common practice to configure your workstation so that AMS starts whenever the machine reboots.

To start AMS on a particular workstation for the first time:

1. If you are starting AMS on the same host as the lock server, skip this step. Otherwise, set up a log directory for AMS log files. Follow the steps in “Setting Up a Log Directory for AMS” on page 28.
2. Start AMS to verify that it is installed properly. To start AMS under the user account `userName`, you can enter a command such as the following:


```
su - userName -c "installDir/arch/bin/oostartams"
```

 AMS will not start if another service already uses the TCP/IP port that AMS expected to use. If you cannot reassign the other service to a different port, you can change the default port for AMS; see Chapter 9, “Advanced Multithreaded Server,” in *Objectivity/DB Administration*.
3. (Optional) Configure your workstation to start AMS whenever the machine reboots. To do this, edit the workstation’s startup script (usually `/etc/rc.local` or `/etc/init.d`) and add a command such as the one in step 2.

By default, AMS handles eight threads. You can specify a different number of threads by entering the `oostartams` command with the `-numthreads` option.

Setting Up a Log Directory for AMS

A running AMS creates log files in a log directory. By default, AMS expects to write log files in the default log directory, which is `/usr/spool/objy`. Alternatively, you can arrange for AMS to write these files into a nondefault log directory.

NOTE If you run AMS and the lock server on the same host, they will both use the same log directory for their log files.

Perform the following steps to set up the log directory *logDir*, where *logDir* could be either the default log directory `/usr/spool/objy` or a nondefault log directory:

1. Log in as `root`.
2. Verify that the desired log directory exists, and, if necessary, create it. Enter:
`mkdir logDir`
3. Set the log directory's permissions to enable AMS to create and update files in this directory, and to enable all users to read and write files there. Enter:
`chmod 777 logDir`

If you set up a nondefault *logDir*, you must enable AMS to find it. To do this:

- ▶ Set the environment variable `OO_SERVER_LOG_DIR` to *logDir* before you start AMS.

Additionally, if you are configuring your workstation to restart AMS upon reboot, you must arrange for the environment variable to be set before AMS server is restarted automatically. You can do this by editing a startup file (such as `.login`) for the *userName* account, and adding a command to set the environment variable.

NOTE AMS will consult the `OO_SERVER_LOG_DIR` environment variable before it looks for the default log directory.

For information on changing the log directory for AMS log files, see “Advanced Multithreaded Server” in *Objectivity/DB Administration*.

Setting Up ootoolmgr

NOTE If Objectivity/Assist is available on your architecture, it is recommended that you use it instead of `ootoolmgr` and skip this section.

Objectivity/DB provides a graphical tool called `ootoolmgr` for browsing objects and types in a database. This tool runs as an X Window System (X) application. Before you can run `ootoolmgr`, you must install several *X resource files* that specify various aspects of tool appearance and behavior. The required files are supplied in subdirectories of *installDir/arch* (the Objectivity/DB installation directory for your architecture).

You can install the required resource files in one of two ways, depending on whether you have access to the standard directory for locating X resource files. This directory is `/usr/lib/X11` on most workstations and `/usr/openwin/lib` on workstations running OpenWindows.

WARNING The X resource files supplied with Objectivity/DB contain resources that are critical to the proper functioning of `ootoolmgr`. *Do not* modify these files.

Installing With Access to /usr/lib/X11

If you have access to the standard X directory, you can install the Objectivity/DB resource files by setting up symbolic links to them.

Caution: You should consult your system administrator before attempting to modify the `/usr/lib/X11` directory or its subdirectories.

To set up the appropriate symbolic links:

1. Log in as `root`, if necessary.
2. Determine whether one or both of the following directories already exists in the file system of each of the workstations where `ootoolmgr` will be run:
 - `/usr/lib/X11/bitmaps`
 - `/usr/lib/X11/app-defaults`

Note: On workstations running OpenWindows, look in `/usr/openwin/lib` instead of `/usr/lib/X11`.

3. Create the `bitmaps` and `app-default` directories, if necessary. Enter:


```
cd /usr/lib/X11
mkdir bitmaps
mkdir app-defaults
```
4. Link the Objectivity/DB resource files to the corresponding X subdirectories. Enter:

```
cd /usr/lib/X11/bitmaps
ln -s installDir/arch/etc/bitmaps/OoToolMgr OoToolMgr
cd /usr/lib/X11/app-defaults
ln -s installDir/arch/etc/app-defaults/OoToolMgr OoToolMgr
```

Installing Without Access to /usr/lib/X11

If you cannot alter the /usr/lib/X11 directory, you can install the Objectivity/DB resource files by setting environment variables. To do this:

1. Set the following environment variables to enable X to find the Objectivity/DB resources. If you are using the C shell, add the following lines to your login files (.cshrc, .login, and so on) using a text editor:

```
setenv XFILESEARCHPATH installDir/arch/etc/app-defaults/%N
setenv XBMLANGPATH installDir/arch/etc/bitmaps/%N/%B
```
2. Update your environment to enable the resource path variables. For example, if you are using the C shell, enter:

```
source homeDir/.cshrc
```

Troubleshooting Installation

This section describes problems that you may encounter when you install Objectivity products and suggests ways to resolve them.

ooverify: File or Directory Existence Errors

Error messages produced by ooverify:

```
Missing files
Missing directories (contents not checked)
Files that should be directories
Directories that should be files
```

Causes: After the product release files are installed onto your disk, ooverify finds that one or more files or directories do not exist in the installation directory.

Solution: Delete the files and directories in the installation area, and reload the Objectivity/DB files from the distribution CD. If the problem persists, your distribution CD may have been damaged. Contact Objectivity Customer Support for help.

ooverify: File or Directory Content Errors

Error messages produced by ooverify:

```
Files of the wrong size
Files with the wrong checksum
Required directories not listed in the manifest
Duplicate entries in the manifest file
Internal check
Bad format in manifest file, line...
```

Causes: After the product release files are installed onto your disk, ooverify finds that the file and directory names in your installation directory match those on the distribution CD; however, there are problems with the contents of the files or the organization of the directories.

Solution: Delete the files and directories in the installation area, and reload the Objectivity/DB files from the distribution CD. If the problem persists, your distribution CD may be incorrect. Contact Objectivity Customer Support for assistance.

ooverify: Access Permissions or File System Errors

Error messages produced by ooverify:

```
Files or directories that cannot be checked (bad symlinks?)
Directories cannot be searched
Files cannot be read (to check checksum)
Can't open manifest file
```

Causes: After the product release files are installed onto your disk, ooverify finds problems with file permission settings or with the file system itself.

Solutions: Check to make sure that you have not changed the permission settings of the files and directories that you created in the Objectivity/DB installation area. If you made changes, use `chmod` to correct the permission settings.

If file or directory permission and file system errors seem to appear intermittently, this may indicate file system or network failure, or workstation problems. Check with other users and the system administrator to make sure that your system is running correctly.

Alternatively, delete the files and directories in the installation area, and reload the Objectivity/DB files from the distribution CD. If the problem persists, you may be following the installation procedure incorrectly. Contact Objectivity Customer Support for assistance.

Lock Server: Directory Permission or File System Errors

Error message produced when starting the lock server:

```
ools:Error Directory /usr/spool/objy not found
O.S. errno = 13 -- permission denied
Failed to connect to server...
```

Cause: A nondefault log directory has not been set up and Objectivity/DB cannot find or access the default log directory `/usr/spool/objy`.

Solutions:

- Check whether the default log directory exists. If it doesn't exist, either create it or set up a nondefault log directory. Follow the steps in "Setting Up a Log Directory for the Lock Server" on page 25.
- If the default directory does exist, either use `chmod` to set less restrictive access privileges for it or start the lock server as the `root` user.
Alternatively, if you are unable to change permissions or start the lock server as the `root` user, you can set up a nondefault log directory. Follow the steps in "Setting Up a Log Directory for the Lock Server" on page 25.

Error message produced when starting the lock server:

```
Warning: Unable to open dirName.
```

Cause: The `OO_SERVER_LOG_DIR` environment variable has been set to a nondefault log directory `dirName`, but Objectivity/DB cannot find or access `dirName`.

Solutions:

- Check whether `dirName` exists. If it doesn't exist, either create it or else set the `OO_SERVER_LOG_DIR` environment variable to refer to a different existing directory. Follow the steps in "Setting Up a Log Directory for the Lock Server" on page 25.
- If the `dirName` does exist, use `chmod` to set less restrictive access privileges for it or start the lock server as the `root` user.
Alternatively, if you are unable to change permissions or start the lock server as the `root` user, you can set up a nondefault log directory. Follow the steps in "Setting Up a Log Directory for the Lock Server" on page 25.

Error message produced when starting the lock server:

```
Not enough disk space.
```

Solution: Free up space on your disk.

Objectivity/C++ Installation

This chapter describes the requirements and steps for installing Objectivity/C++ on a UNIX platform. You can install Objectivity/C++ with or without the Objectivity/C++ Data Definition Language (Objectivity/DDL) option.

- Objectivity/C++ is a programming interface for writing C++ applications that store and manipulate persistent data in a federated database.
- Objectivity/DDL is a preprocessor for converting Data Definition Language (DDL) files into a schema of data types in a federated database. The DDL processor also produces source and header files for these data types.

System Requirements

You can install Objectivity/C++ on any of the UNIX architectures listed in Table 1-1 on page 14.

Software Requirements

Objectivity/C++ requires that the following software be installed:

- Objectivity/DB (see Chapter 1)
 - The vendor-supplied C++ compiler for your architecture
- Note:** See the release information on the Objectivity Technical Support Web site for the currently supported C++ compiler versions. Contact Objectivity Customer Support to get access to this Web site.

You can install Objectivity/C++ without Objectivity/DDL. However, you cannot install Objectivity/DDL without also installing Objectivity/C++.

Installing Objectivity/C++ or Objectivity/DDL

To install Objectivity/C++, Objectivity/DDL, or both:

1. Log in to your workstation as `root`.
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 35).
3. If you have not done so already, obtain the Objectivity installation files as described in “Accessing Objectivity Installation Files” on page 15.
4. Run the Objectivity installation script with a command such as the following:

```
./install.sh
```
5. When prompted, select **Custom Installation** and select the product or products to be installed:
 - To install only Objectivity/C++, enter its item number.
 - To install Objectivity/C++ and Objectivity/DDL, enter the corresponding item numbers, separated by commas.

You cannot install Objectivity/DDL without Objectivity/C++.

NOTE You must be licensed for every product you install.

6. At the directory prompt, specify the directory in which Objectivity/DB is installed (*installDir*). If you get an error message reporting insufficient disk space or incorrect permissions, press Enter to exit the installation script; then fix the problem and restart the installation script (see step 4).
7. At the `ooconfig` prompts (Objectivity/DDL only), specify the requested compiler and environment information or accept the defaults. The `ooconfig` script configures the DDL processor and creates the DDL processor executable (`ooddlx`).

The installation script:

- Places the product release files in subdirectories of *installDir/arch*, where *arch* is the architecture name for your platform.
- Runs the installation verification test `ooverify`. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 31.

Preparing to Use Objectivity/C++ or Objectivity/DDL

After the installation script completes, perform the following steps:

1. Familiarize yourself with your installation by reading “Release Files for Objectivity/C++ and Objectivity/DDL” on page 38.
2. (Optional) Test the installation of Objectivity/C++ and Objectivity/DDL by running the provided C++ sample application; see “Testing Objectivity/C++ Setup” on page 38.
3. (Optional) If Objectivity/Assist is available on your architecture, use it to perform the C++ Tutorial for getting started with Objectivity/C++ application development.
4. Read:
 - Appendix A, “C++ Application Development,” in this book for platform-specific information about using Objectivity/C++.
 - *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the file `installDir/arch/doc/ReleaseNotes.pdf`.
 - The release information on the Objectivity Technical Support Web site for known problems and currently supported C++ compiler versions. Contact Objectivity Customer Support to get access to this Web site.

Release Files for Objectivity/C++ and Objectivity/DDL

When you install Objectivity/C++ and Objectivity/DDL, their files are organized in subdirectories of the *installDir/arch* directory, as shown in Table 2-1.

Table 2-1: Release Files in *installDir/arch*

Subdirectory	Contains
bin	Objectivity/DDL script (<i>ooconfig</i>) for configuring the DDL processor; executable (<i>ooddlx</i>) created by <i>ooconfig</i>
doc	PDF files for online books: <i>Objectivity/C++ Data Definition Language</i> , <i>Objectivity/C++ Programmer's Guide</i> , and <i>Objectivity/C++ Programmer's Reference</i>
include	Include files for the C++ programming interface
lib	Static and shared libraries for linking C++ applications ^a
	Shared library for persistent collections ^b
samples/cxx	Sample C++ database applications for demonstration and testing

a. See "Linking Applications to Objectivity/DB" on page 99.

b. See "Linking to Additional Objectivity Products and Features" on page 104.

The installation directory structure is the same for all architectures, which means you can develop applications that will compile on other platforms without having to consider workstation-specific directory naming conventions.

Testing Objectivity/C++ Setup

If you installed both Objectivity/C++ and Objectivity/DDL, you can test whether they are set up correctly by building and running the HelloWorld sample application provided with the installation.

When you build the sample application, the makefile creates a federated database. Executing the sample application populates the federated database with persistent objects.

You can also inspect the various sample applications in the directory *installDir/arch/samples/cxx* to see how to use various Objectivity/C++ features; see "Overview of Sample Applications" on page 118.

Building and Executing the Sample Application

To build and run the C++ sample application, follow these steps:

1. If you have not done so already, set up a default license file for your current Objectivity license; see “Setting Up a License File” on page 21.
2. If necessary, start the lock server; see “Setting Up the Lock Server” on page 24. For example, to start the lock server on the current host, enter:

```
installDir/arch/bin/oolockserver
```

3. Copy the *installDir/arch/samples/cxx/helloWorld* directory. Then give yourself read, write, and execute permissions to the copied directory, and change your working directory to this new location.

For example, enter the following, where *homeDir* represents your home directory:

```
cp -r installDir/arch/samples/cxx/helloWorld homeDir
chmod 777 homeDir/helloWorld
cd homeDir/helloWorld
```

4. Copy the *installDir/arch/samples/cxx/include.mk* file to the directory containing the *helloWorld* directory.

For example:

```
cp installDir/arch/samples/cxx/include.mk ..
```

5. Set the following environment variables as appropriate to your installation:
 - If you installed Objectivity/DB in a location other than the default directory (*/opt/object*), set *OBJY_HOME* to the location of the Objectivity/DB installation directory *installDir* (not including *arch*)—for example:


```
setenv OBJY_HOME /usr/myObjy
```
 - If your lock server is not running on the current host, set *OBJY_LS* to the name of the lock-server host—for example:


```
setenv OBJY_LS myLockServerHost
```
6. Set the following environment variables to specify the federated database to be created by make:
 - Set *OBJY_FDPATH* to specify the path for the directory to contain the boot file *HelloWorld.boot*—for example:


```
setenv OBJY_FDPATH homeDir/helloWorld
```
 - Set *OBJY_FDID* to a unique federated database identifier (a number from 1 to 32,000)—for example:


```
setenv OBJY_FDID 4567
```

7. Create the demo federated database and build the sample application (helloWorld). To do so, enter:
`make -e`
The `-e` flag causes `make` to use the environment variables you set.
8. Run the sample application. Enter:
`helloWorld HelloWorld.boot`

Demo Results

If Objectivity/C++ is set up correctly, the sample application displays:

```
Welcome to the Objectivity HelloWorld Example
Using FD Bootfile passed as argument: HelloWorld.boot
Create database
Create HelloObject and set message
Looking for object with name HelloWorld
Found it!: HelloWorld
The HelloWorld test has PASSED.
```

If Objectivity/C++ is not set up correctly, the sample application displays:

```
The HelloWorld test has FAILED.
```

If other messages are displayed, check the compiler, the lock server, and the settings of the environment variables. Correct any errors and run the sample applications again. If there are no installation, `make`, or compiler errors, and the applications still fails, contact Objectivity Customer Support for assistance.

Objectivity/DB Active Schema for C++ Installation

This chapter describes the requirements and steps for installing the C++ programming interface to Objectivity/DB Active Schema (Active Schema) on a UNIX platform. Active Schema for C++ enables you to write C++ applications that dynamically read and modify the schemas in Objectivity/DB federated databases.

System Requirements

You can install Active Schema for C++ on any of the UNIX architectures listed in Table 1-1 on page 14.

Software Requirements

Active Schema for C++ requires that the following software be installed on your system:

- Objectivity/DB (see Chapter 1)
- Objectivity/C++ and Objectivity/DDL (see Chapter 2)

Installing Active Schema for C++

To install Active Schema for C++:

1. Log in to your workstation as `root`.
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 41).
3. If you have not done so already, obtain the Objectivity installation files as described in “Accessing Objectivity Installation Files” on page 15.
4. Run the Objectivity installation script with a command such as the following:

```
./install.sh
```

5. When prompted, select **Custom Installation**; then select Active Schema for C++ from the displayed list.

NOTE You must be licensed for every product you install.

6. At the directory prompt, specify the directory in which Objectivity/DB is installed (*installDir*). If you get an error message reporting insufficient disk space or incorrect permissions, press Enter to exit the installation script; fix the problem and restart the installation script (see step 4).

The installation script:

- Places the release files in subdirectories of *installDir/arch*, where *arch* is the architecture name for your platform.
- Runs the installation verification test *ooverify*. If *ooverify* displays error messages, see “Troubleshooting Installation” on page 31.

Preparing to Use Active Schema for C++

After the installation script completes, perform the following steps:

1. Familiarize yourself with your installation by reading “Release Files for Active Schema for C++” on page 43.
2. Read:
 - Appendix A, “C++ Application Development,” in this book for platform-specific information about compiling and linking this product.
 - *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the file *installDir/arch/doc/ReleaseNotes.pdf*.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

Release Files for Active Schema for C++

When you install Active Schema for C++, its files are organized in subdirectories of the *installDir/arch* directory, as shown in Table 3-1.

Table 3-1: Release Files in *installDir/arch*

Subdirectory	Contains
doc	PDF file for the online book Objectivity/DB Active Schema for C++
include	Include file <code>ooas.h</code> for the C++ programming interface to Active Schema
lib	Shared link libraries for Active Schema ^a

a. See “Linking to Additional Objectivity Products and Features” on page 104.

Objectivity for Java Installation

This chapter describes the requirements and steps for installing Objectivity for Java on a UNIX platform. Objectivity for Java is a programming interface for writing Java applications that store and manipulate persistent data in an Objectivity/DB federated database.

System Requirements

You can install Objectivity for Java on any of the UNIX architectures listed in Table 1-1 on page 14.

If you are using Objectivity for Java with Java Connection Architecture (JCA), you must use one of the following UNIX architectures: `linux86_64`, `linux86gcc3`, `solaris7`, `solaris86_64`, or `sparc64`.

Software Requirements

Objectivity for Java requires that the following software be installed:

- Objectivity/DB (see Chapter 1)
- The supported Java Development Kit (JDK) for your architecture.
See the release information on the Objectivity Technical Support Web site for the currently supported JDK versions. Contact Objectivity Customer Support to get access to this Web site.
Note: See “Finalizing Objects in JDK 1.6” on page 113 for information about a workaround for a JDK defect that affects persistence-capable classes with finalizers.
- A Web browser to view Objectivity for Java online books in HTML format.

The Java Connection Architecture (JCA) package of Objectivity for Java requires that the BEA WebLogic Server 8.1 application server be installed.

Installing Objectivity for Java

To install Objectivity for Java:

1. Log in to your workstation as `root`.
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 45).
3. If you have not done so already, obtain the Objectivity installation files as described in “Accessing Objectivity Installation Files” on page 15.
4. Run the Objectivity installation script with a command such as the following:

```
./install.sh
```
5. When prompted, select **Custom Installation**; then select Objectivity for Java from the displayed list.

NOTE You must be licensed for every product you install.

6. At the directory prompt, specify the directory in which Objectivity/DB is installed (*installDir*). If you get an error message reporting insufficient disk space or incorrect permissions, press Enter to exit the installation script; then fix the problem and restart the installation script (see step 4).

The installation script:

- Places the release files in subdirectories of *installDir/arch*, where *arch* is the architecture name for your platform.
- Runs the installation verification test `ooverify`. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 31.

Preparing to Use Objectivity for Java

After the installation script completes, perform the following steps:

1. Familiarize yourself with your installation by reading “Release Files for Objectivity for Java” on page 48.
2. Set the CLASSPATH environment variable as follows:

```
setenv CLASSPATH
    installDir/arch/java/lib/ojjava.jar:existingValues
```

where *existingValues* represents existing class path components.
For some application-development environments, you must specify CLASSPATH from within the tool.
3. (Optional) Test your installation by running the provided sample application; see “Testing Objectivity for Java Setup” on page 48.
4. (Optional) If Objectivity/Assist is available on your architecture, use it to perform the Java Tutorial for getting started with Objectivity for Java application development.
5. If you are using Objectivity for Java with the JCA package, perform the steps in the package description for `com.objy.jca` in the online HTML book, *Objectivity for Java Programmer's Reference*.
6. Read:
 - Appendix B, “Java Application Development,” in this book for platform-specific information about using Objectivity for Java.
 - *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the file `installDir/arch/doc/ReleaseNotes.pdf`.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

Release Files for Objectivity for Java

The Objectivity for Java installation script places files in subdirectories of the *installDir/arch* directory, where *arch* is the architecture name for your platform, as shown in Table 4-1.

Table 4-1: Objectivity for Java Release Files in *installDir/arch*

Directory	Contains	
doc	javaGuide.pdf	Online guide (PDF)
	java/index.html	Document index
	java/api/.../*.html	Online reference
	java/guide/*.html	Online guide (HTML)
	java/samples/**/*.java java/samples/**/*.java	Java database applications used as examples in the guide
java	src/src.zip	Library source
	lib/oojava.jar	Library executable
lib	liboojava.so	Library executable
samples	java/**/*.java	Sample Java database applications for demonstration and testing

Testing Objectivity for Java Setup

You can test whether Objectivity for Java is set up correctly by building and running the HelloWorld sample application provided with the installation. You can build this sample application either using a Java IDE or command-line tools. Executing the sample application populates an Objectivity/DB federated database with persistent objects.

You can inspect the various sample applications in subdirectories of the directory *installDir/arch/samples/java* to see how to use various Objectivity for Java features. (You can also inspect the applications in the directory *installDir/arch/doc/java/samples*, which are used as part of the Objectivity for Java documentation.)

To build and run the sample application using command-line tools:

1. If you have not already done so, set up a default license file for your current Objectivity license; see “Setting Up a License File” on page 21.
2. If necessary, start the lock server; see “Setting Up the Lock Server” on page 24. For example, to start the lock server on the current host, enter:

```
installDir/arch/bin/oolockserver
```

3. Copy the `installDir/arch/samples/java/helloWorld` directory. Then give yourself read, write, and execute permissions to the copied directory. For example, enter the following, where `homeDir` represents your home directory:

```
cp -r installDir/arch/samples/java/helloWorld homeDir
chmod 777 homeDir/helloWorld
```

4. Change your working directory to the `src` subdirectory of the new location. For example, enter the following, where `homeDir` represents your home directory:

```
cd homeDir/helloWorld/src
```

5. Create the HelloWorld federated database. Enter:

```
oonewfd -fdfilepath HelloWorld.fdb
        -lockserverhost hostName HelloWorld.boot
```

where `hostName` is the name of the lock-server host.

For more information about creating a federated database, see Chapter 4, “Federated Database Tasks,” of *Objectivity/DB Administration*.

6. Build the sample application. Enter:

```
javac *.java
```

7. Run the sample application. Enter:

```
java Main HelloWorld.boot
```

If Objectivity for Java is set up correctly, the sample application displays messages such as:

```
Welcome to the Objectivity HelloWorld Example
Using FD Bootfile passed as argument: HelloWorld.boot
Create database
Create HelloObject and set message
Looking for object with name HelloWorld
Found it!: HelloWorld
The HelloWorld test has PASSED.
```

If Objectivity for Java is not set up correctly, the sample application either produces a Java runtime error or displays the following message:

```
The HelloWorld test has FAILED.
```

If the installation test failed:

- 1.** Verify your environment-variable settings and default Objectivity license file.
- 2.** Correct any errors.
- 3.** Rebuild and rerun the application.

If the application still fails, contact Objectivity Customer Support for assistance.

Objectivity/DB Active Schema for Java Installation

This chapter describes the requirements and steps for installing the Java programming interface to Objectivity/DB Active Schema (Active Schema) on a UNIX platform. Active Schema for Java enables you to write Java applications that dynamically read and modify the schemas in Objectivity/DB federated databases.

NOTE Active Schema for Java is automatically installed when you install Objectivity for Java. You can use Active Schema for Java only if your Objectivity license authorizes it.

System Requirements

You can use Active Schema for Java on any of the UNIX architectures listed in Table 1-1 on page 14.

See the release information on the Objectivity Technical Support Web site for the currently supported architectures. Contact Objectivity Customer Support to get access to this Web site.

Software Requirements

Active Schema for Java requires that the following software be installed on your system:

- Objectivity/DB (see Chapter 1)
- Objectivity for Java (see Chapter 4)

Installing Active Schema for Java

To install Active Schema for Java:

- If you have not already done so, perform the steps for installing Objectivity for Java; see “Installing Objectivity for Java” on page 46.

Preparing to Use Active Schema for Java

After the installation script completes, perform the following steps:

1. If you have not already done so, perform the steps for setting up Objectivity for Java; see “Preparing to Use Objectivity for Java” on page 47.
2. If your Objectivity license does not authorize you to use Active Schema for Java, contact your account manager to purchase appropriate licensing.
3. With an HTML browser, familiarize yourself with the product documentation; see Table 5-1.
4. Read:
 - Appendix B, “Java Application Development,” in this book for any platform-specific information about using Active Schema for Java.
 - *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the file `installDir/arch/doc/ReleaseNotes.pdf`.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

Release Files for Active Schema for Java

Active Schema for Java software is incorporated in the release files of Objectivity for Java, so Active Schema for Java does not have its own separate release files.

The online reference for Active Schema for Java is incorporated in the Objectivity for Java Programmer’s Reference, and is located within the `installDir/arch` directory, as shown in Table 5-1.

Table 5-1: Active Schema for Java Documentation in `installDir/arch`

Directory	Files	Contains
doc	<code>java/index.html</code>	Document index
	<code>java/api/com/objy/as/.../*.html</code>	Online reference

Objectivity/Smalltalk for VisualWorks Installation

This chapter describes the requirements and steps for installing Objectivity/Smalltalk for VisualWorks on a UNIX platform. Objectivity/Smalltalk is a programming interface for writing Smalltalk applications that store and manipulate persistent data in an Objectivity/DB federated database.

System Requirements

You can install Objectivity/Smalltalk for VisualWorks on any of the UNIX architectures listed in Table 1-1 on page 14, *except* hpuxia64, linux86_64, solaris86_64, and sparc64.

Software Requirements

Objectivity/Smalltalk for VisualWorks requires that the following software be installed on your system:

- Objectivity/DB (see Chapter 1)
- Cincom VisualWorks
- (Optional) OTI ENVY/Developer

Note: See the release information on the Objectivity Technical Support Web site for the currently supported versions of VisualWorks and ENVY/Developer. Contact Objectivity Customer Support to get access to this Web site.

Installing Objectivity/Smalltalk for VisualWorks

To install Objectivity/Smalltalk for VisualWorks:

1. Log in to your workstation as `root`.
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 53).
3. If you have not done so already, obtain the Objectivity installation files as described in “Accessing Objectivity Installation Files” on page 15.
4. Run the Objectivity installation script with a command such as the following:

```
./install.sh
```
5. When prompted, select **Custom Installation**; then select Objectivity/Smalltalk for VisualWorks from the displayed list.

NOTE You must be licensed for every product you install.

6. At the directory prompt, specify the directory in which Objectivity/DB is installed (*installDir*). If you get an error message reporting insufficient disk space or incorrect permissions, press Enter to exit the installation script; fix the problem and restart the installation script (see step 4).

The installation script:

- Places the release files in subdirectories of *installDir/arch*, where *arch* is the architecture name for your platform.
- Runs the installation verification test `ooverify`. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 31.

Preparing to Use Objectivity/Smalltalk for VisualWorks

After the installation script completes, perform the following steps:

1. Familiarize yourself with your installation by reading “Release Files for Objectivity/Smalltalk for VisualWorks” on page 55.
2. Set up your image by performing the steps in one of the following sections:
 - “Setting Up VisualWorks” on page 55
 - “Setting Up VisualWorks With ENVY/Developer” on page 56
3. Read:
 - *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the file *installDir/arch/doc/ReleaseNotes.pdf*.

- The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

Release Files for Objectivity/Smalltalk for VisualWorks

The Objectivity/Smalltalk for VisualWorks installation script installs various files in subdirectories of the `installDir/arch` directory, as shown in Table 6-1.

Table 6-1: Objectivity/Smalltalk Release Files in `installDir/arch`

Subdirectory	File	Description
doc	smalltalk.pdf	PDF file for the online book <i>Objectivity/Smalltalk for VisualWorks</i>
lib	<i>Filenames differ on each platform</i>	Objectivity/Smalltalk shared library (bridge to Objectivity/DB kernel)
etc/smalltalk	objyDB.pcl	External interface class required for developing and deploying applications in VisualWorks
	objyDB.pst	Parcel source file for Objectivity/Smalltalk for VisualWorks
	objyDB.dat	ENVY/Developer repository containing Objectivity/Smalltalk for VisualWorks applications
	objyTHRD.st	Objectivity/Smalltalk threadsafe option file-in for VisualWorks. <i>Do not</i> file in this file unless you plan to use the threadsafe option; see <i>Objectivity/Smalltalk for VisualWorks</i> for details.

Setting Up VisualWorks

NOTE If you use VisualWorks with ENVY/Developer, go to the next section instead.

To set up VisualWorks for use with Objectivity/Smalltalk:

1. Start VisualWorks with a fresh image.

2. Parcel in the file:

```
installDir/arch/etc/smalltalk/objyDB.pcl
```

3. Save the image.
4. File in your development code.
5. (Optional) Delete the file `installDir/arch/etc/smalltalk/objyDB.dat` to free up disk space. This file is only used with ENVY/Developer.
6. (Optional) Test the Objectivity/Smalltalk installation (see “Testing Objectivity/Smalltalk for VisualWorks Setup” on page 56).

Setting Up VisualWorks With ENVY/Developer

To set up ENVY/Developer for use with Objectivity/Smalltalk:

1. Start VisualWorks with a fresh ENVY/Developer image.
2. Open a **Configuration Maps Browser**.
3. Import all of the configuration maps into your ENVY server repository from `installDir/arch/etc/smalltalk/objyDB.dat`.

When you attempt to import from the **Configuration Maps Browser**, remember that ENVY/Developer will prevent accessing a file that is not local to the machine running `emsrv`, unless `emsrv` was started using the `-xn` option.

4. Use the option **load with required maps** for the configuration map **Objectivity/DB**.
5. Save the image.
6. File in your development code.
7. (Optional) Test the product installation on ENVY/Developer (see “Testing Objectivity/Smalltalk for VisualWorks Setup” below).
8. (Optional) After importing the configuration maps and sample application code (see *Objectivity/Smalltalk for VisualWorks*) from `objyDB.dat`, delete this file from your system to free disk space.

Testing Objectivity/Smalltalk for VisualWorks Setup

You can test whether Objectivity/Smalltalk for VisualWorks is set up correctly. To do so:

- In your VisualWorks image, evaluate the expression:


```
OoReleaseInstallUtility verifyInstall
```

 This method sends output to the Transcript.

You can test the basic functionality of Objectivity/Smalltalk for VisualWorks by performing the following steps:

1. Set up a default license file for your current Objectivity license, if you have not already done so; see “Setting Up a License File” on page 21.
2. Create a federated database; see Chapter 4, “Federated Database Tasks,” of *Objectivity/DB Administration*.
3. In your VisualWorks image, evaluate the following expression:
`OoReleaseInstallUtility verifyInstall: bootFilePath`
where *bootFilePath* is the path to the boot file of a federated database.

Objectivity/Python Installation

This chapter describes the requirements and steps for installing Objectivity/Python on a UNIX platform. Objectivity/Python is a programming interface for writing Python applications and scripts that store and manipulate persistent data in an Objectivity/DB federated database.

System Requirements

You can install Objectivity/Python on any of the UNIX architectures listed in Table 1-1 on page 14.

Software Requirements

Objectivity/Python requires that the following software be installed on your system:

- Objectivity/DB (see Chapter 1)
- The version of Python compatible with the current release of Objectivity/DB

Note: See the Objectivity Technical Support Web site for the compatible version of Python for the current release of Objectivity/DB. Contact Objectivity Customer Support to get access to this Web site.

On certain architectures you will need to configure your software to work with the python executable. See “Setting Up Python” on page 62.

Installing Objectivity/Python

To install Objectivity/Python:

1. Log in to your workstation as `root`.
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” above).
3. If you have not done so already, obtain the Objectivity installation files as described in “Accessing Objectivity Installation Files” on page 15.
4. Run the Objectivity installation script with a command such as the following:

```
./install.sh
```
5. When prompted, select **Custom Installation**; then select Objectivity/Python from the displayed list.

NOTE You must be licensed for every product you install.

6. At the directory prompt, specify the directory in which Objectivity/DB is installed (*installDir*). If you get an error message reporting insufficient disk space or incorrect permissions, press Enter to exit the installation script; then fix the problem and restart the installation script (see step 4).

The installation script:

- Places the release files in subdirectories of *installDir/arch*, where *arch* is the architecture name for your platform.
- Runs the installation verification test `ooverify`. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 31.

Preparing to Use Objectivity/Python

After the installation script completes, perform the following steps:

1. Familiarize yourself with your installation by reading “Release Files for Objectivity/Python” on page 61.
2. Set the PYTHONPATH environment variable as follows:


```
setenv PYTHONPATH installDir/arch/lib
```
3. Check your shared library search path. If it is not set, see step 4 in “Preparing to Use Objectivity/DB” on page 18.
4. On selected architectures, set up the python executable to be compatible with Objectivity/DB, if you have not already done so. See “Setting Up Python” below.
5. Read:
 - *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the file *installDir/arch/doc/ReleaseNotes.pdf*.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

Release Files for Objectivity/Python

When you install Objectivity/Python, its files are organized in subdirectories of the *installDir/arch* directory, as shown in Table 7-1.

Table 7-1: Objectivity/Python Release Files in *installDir/arch*

Subdirectory	Contains	
doc	python/index.html	Document index
	python/api/*.html	<i>Objectivity/Python Programmer's Reference</i> (HTML)
lib	Shared library for linking Objectivity/Python applications	
samples	python/**/*.py	Sample Objectivity/Python database applications for demonstration and testing.

Setting Up Python

The python executable does not natively interoperate with the following architectures: `hprisc`, `hpuxia64`, `solaris86_64`, and `sparc64`. To develop applications on these architectures with Objectivity/Python, you must customize your python executable:

- On `hprisc` and `hpuxia64`, you must link the `pthread` library into the python executable; see “Customizing Python on `hprisc` and `hpuxia64`” on page 62.
- On `solaris86_64` and `sparc64`, you must customize the python executable so that it works with 64-bit architectures; see “Customizing Python on `solaris86_64` and `sparc64`” on page 65.

In addition, on `hprisc`, and `hpuxia64`, you must increase the default thread stack size.

Customizing Python on `hprisc` and `hpuxia64`

The Objectivity/Python shared library (module) uses the shared `pthread` library and therefore contains Thread Local Storage (TLS); however, the `hprisc` and `hpuxia64` architectures do not natively interoperate with shared libraries that contain TLS. Consequently, for interoperation to occur, you must link the shared `pthread` library (and dependencies) into the python executable, so that the libraries are pre-loaded when the Objectivity/Python module is dynamically loaded.

NOTE On `hpuxia64`, the python executable must be built in 64-bit mode. Additional information on building `hpuxia64` can be found in the Python README file which is included with your Python source distribution.

As part of customizing the python executable on the `hprisc` and `hpuxia64` architectures, the default thread stack size used by the python executable must be increased. The ideal value depends upon your application’s memory needs. An initial value of `0x50000` is recommended for most applications.

On `hprisc`

1. Set environment variables as follows. Enter the following at a command prompt:

```
setenv BASECFLAGS "-DTHREAD_STACK_SIZE=0x50000"  
setenv LDFLAGS "-L."
```

2. Run the configure script by entering the following at a command prompt:


```
./configure --without-gcc --enable-shared \
  --with-libs="-lpthread -lstd_v2 -lCsup_v2 -lcl"
```
3. Enter the following at a command prompt:


```
make
make test
make install
```

Workarounds for make errors on hprisc

Some versions of the HP-UX C compiler cause the `make` command to fail when the debug flag (`-g`) is used. If you encounter such a failure, you can:

- Edit the makefile produced by step 2 above, and remove the `-g` compiler option.

Some versions of the HP-UX C compiler cause the `make` command to fail on certain syntax within a Python source file. If you encounter such a failure, you can:

1. Open the source file `Python/Python-ast.c`.
2. Modify the `PyAST_obj2mod(...)` function as follows (line numbers may vary):
 - Delete or comment out the following lines:

```
6735: PyObject *req_type[] = {(PyObject*)Module_type, (PyObject*)Expression_type,
6736:                          (PyObject*)Interactive_type};
```

- Replace the following line:

```
6743: isinstance = PyObject_IsInstance(ast, req_type[mode]);
```

with these lines:

```
if(mode == 0)
    isinstance = PyObject_IsInstance(ast, (PyObject*)Module_type);
if(mode == 1)
    isinstance = PyObject_IsInstance(ast, (PyObject*)Expression_type);
if(mode == 2)
    isinstance = PyObject_IsInstance(ast, (PyObject*)Interactive_type);
```

On hpuxia64

1. Set environment variables as follows. Enter the following at a command prompt:

```
setenv BASECFLAGS "+DD64 -DTHREAD_STACK_SIZE=0x50000"  
setenv LDFLAGS "-L/usr/lib/hpux64 -L."  
setenv CC cc  
setenv CXX aCC
```
2. Run the configure script by entering the following at a command prompt:

```
./configure --without-gcc --enable-shared \  
--with-libs="-lpthread -lCsup -lstd_v2 -lunwind"
```
3. Unset environment variables as follows. Enter the following at a command prompt:

```
unsetenv BASECFLAGS  
unsetenv LDFLAGS  
unsetenv CC  
unsetenv CXX
```
4. Edit the makefile as follows:
 - Remove `-O` from the `OPT` line
 - Remove `-Ae` from the `CC` line.
 - Remove `-Olimit 1500` from the `BASECFLAGS` line.
5. Enter the following at a command prompt:

```
make  
make test  
make install
```

Customizing Python on solaris86_64 and sparc64

You must customize your `solaris86_64` build and your `sparc64` build of the `python` executable with the necessary options for use with Objectivity/Python on 64-bit architectures.

On solaris86_64

1. Set environment variables as follows. At a command prompt, enter:

```
setenv BASECFLAGS "-xarch=amd64"  
setenv LDFLAGS "-xarch=amd64"  
setenv CC "-xarch=amd64"
```
2. Run the `configure` script by entering the following at a command prompt:

```
./configure --without-gcc --enable-shared
```
3. Enter the following at a command prompt:

```
make  
make test  
make install
```

On sparc64

NOTE An installation of the 64-bit version of the Tcl/Tk package is required for building the `python` executable on `sparc64`.

1. Set environment variables as follows. At a command prompt, enter:

```
setenv BASECFLAGS "-xarch=v9"  
setenv LDFLAGS "-xarch=v9"
```

Note: Some versions of the compiler will accept the `-m64` flag instead of the `-xarch=v9` flag.
2. Run the `configure` script by entering the following at a command prompt:

```
./configure
```
3. Enter the following at a command prompt:

```
make  
make test  
make install
```


Objectivity/SQL++ Installation

This chapter describes the requirements and steps for installing Objectivity/SQL++ on a UNIX platform. Objectivity/SQL++ provides ANSI-standard SQL-92 access to objects in an Objectivity/DB federated database, with extensions to accommodate the Objectivity/DB object model.

Objectivity/SQL++ has the following two components:

- The Objectivity/SQL++ ODBC server, a process that enables ODBC-compliant applications to access Objectivity/DB federated databases. (Requires the separately installed Objectivity/SQL++ ODBC Driver.)
- Interactive SQL++, a tool for interactively submitting SQL statements or scripts to an Objectivity/DB federated database.

Objectivity/SQL++ embeds Dharma/SQL 7.x and therefore implements the ODBC 3.0 database-access standard.

System Requirements

You can install Objectivity/SQL++ on any of the UNIX architectures listed in Table 1-1 on page 14.

Software Requirements

At a minimum, Objectivity/SQL++ requires that the following software be installed on your system:

- Objectivity/DB (see Chapter 1)

The following additional software is required for building and running the sample applications that verify Objectivity/SQL++ installation:

- Objectivity/C++ and Objectivity/DDL (see Chapter 2)

The following additional software is required when using the ODBC server:

- Objectivity/SQL++ ODBC Driver

Installing Objectivity/SQL++

To install Objectivity/SQL++:

1. Log in to your workstation as `root`.
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” above).
3. If you have not done so already, obtain the Objectivity installation files as described in “Accessing Objectivity Installation Files” on page 15.
4. Run the Objectivity installation script with a command such as the following:

```
./install.sh
```
5. When prompted, select **Custom Installation**; then select Objectivity/SQL++ from the displayed list.

NOTE You must be licensed for every product you install.

6. At the directory prompt, specify the directory in which Objectivity/DB is installed (*installDir*). If you get an error message reporting insufficient disk space or incorrect permissions, press Enter to exit the installation script; then fix the problem and restart the installation script (see step 4).

The installation script:

- Places the release files in subdirectories of *installDir/arch*, where *arch* is the architecture name for your platform.
- Runs the installation verification test `ooverify`. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 31.

Preparing to Use Objectivity/SQL++

After the installation script completes, perform the following steps:

1. Familiarize yourself with your installation by reading “Release Files for Objectivity/SQL++” on page 70.
2. Create an account with the username `sysupe` and a password of your choice. The Objectivity/SQL++ database administrator will use this account. For more information, see *Objectivity/SQL++*.
3. Set the environment variable `OO_SQL_DIR` to the fully qualified pathname of the Objectivity/DB installation directory (*installDir/arch*). Do this for the `sysupe` account—for example, in a startup file such as `.login`—and for each user that will run Interactive SQL++. Objectivity/SQL++ uses `OO_SQL_DIR` to find help files and error-code files.
4. Set up the Objectivity/SQL++ ODBC server by following the steps in “Setting Up the Objectivity/SQL++ ODBC Server” on page 70.
5. Test each Objectivity/SQL++ component you plan to use:
 - Test Interactive SQL++ by following the steps in “Testing Interactive SQL++” on page 73.
 - Set up the ODBC server so you can test it together with an Objectivity/SQL++ ODBC Driver that has been installed in the same TCP/IP network. Follow the steps in “Preparing the ODBC Server for Testing” on page 74.
6. Read:
 - *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the file *installDir/arch/doc/ReleaseNotes.pdf*.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

Release Files for Objectivity/SQL++

When you install Objectivity/SQL++, its files are organized in subdirectories of the *installDir/arch* directory, as shown in Table 8-1. For information about these files, see *Objectivity/SQL++*.

Table 8-1: Objectivity/SQL++ Release Files in *installDir/arch*

Subdirectory	Contains
bin	Executables for Interactive SQL++ and the Objectivity/SQL++ ODBC server
doc	PDF file for the online book <i>Objectivity/SQL++</i>
etc/sql	Subdirectories containing source files for defining new triggers and stored procedures, help files and Interactive SQL++ configuration files, and PDF files for Dharma/SQL documentation
include	Include files for developing Objectivity/SQL++ triggers and stored procedures
lib	Link library for building user-defined triggers and stored procedures
samples/sql	Subdirectories containing applications for demonstration and testing

Setting Up the Objectivity/SQL++ ODBC Server

Objectivity/SQL++ provides an ODBC server, which enables ODBC-compliant applications to connect to a federated database. For information about the ODBC server, see Chapter 3, “Getting Started With ODBC,” in *Objectivity/SQL++*.

You set up the ODBC server by:

- Specifying a port number
- Arranging for startup whenever the machine reboots
- Optionally specifying a nondefault log directory

Specifying the ODBC Server's Port Number

You must associate a port number with the Objectivity/SQL++ ODBC server so that remote ODBC-compliant applications can connect to it through the Objectivity/SQL++ ODBC Driver. To do this:

1. Log in as `root`.
2. Add the following entry to the TCP/IP `services` file. Typically, this file is `/etc/services`.

```
oosqlnw      1990/tcp      # Objectivity/SQL++ Server
```

 If you are using the Network Information Service (NIS), you should ask your system administrator to perform the equivalent operation for your NIS configuration.
3. If another service already uses TCP/IP port 1990, either reassign that service to a different port (recommended) or assign a different port to the `oosqlnw` service.

Important: If you change the TCP/IP port for the Objectivity/SQL++ ODBC server, you must assign the *same* port to the `oosqlnw` service on each Objectivity/SQL++ ODBC Driver host.

Arranging for ODBC-Server Startup

You normally configure your workstation so that the ODBC server starts automatically whenever the machine reboots.

A running ODBC server is required for accessing Objectivity/DB federated databases with ODBC-compliant applications that use the Objectivity/SQL++ ODBC Driver. A running ODBC server is *not* required for using Interactive SQL++.

You must run ODBC server under the `systpe` account. For information about required access permissions, see Chapter 3, "Getting Started With ODBC," in *Objectivity/SQL++*.

Perform the following steps when you start the ODBC server on a particular workstation for the first time:

1. Log in as `root` (required for step 5).
2. Set up a log directory for ODBC log files. Follow the steps in "Setting Up a Log Directory for the ODBC Server" on page 72.
3. Start the ODBC server to verify that it is installed properly. To start the ODBC server under the user account `systpe`, you can enter a command such as the following:

```
su - systpe -c "installDir/arch/bin/oosqld start"
```

4. If the ODBC server does not start, check whether a port conflict exists; see “Specifying the ODBC Server’s Port Number” on page 71.
5. (Recommended) Configure your workstation to start the ODBC server whenever the machine reboots. To do this, edit the workstation’s startup script (usually `/etc/rc.local` or `/etc/init.d`) and add a command such as the one shown in step 3.

Setting Up a Log Directory for the ODBC Server

A running ODBC server creates log files in a log directory. By default, the ODBC server expects to write log files in the default log directory, which is `/usr/spool/objy`. Alternatively, you can choose to write these files into a nondefault log directory.

NOTE If you are starting the ODBC server on the same host as the lock server or AMS and you created `/usr/spool/objy` the default log directory on that host, the ODBC server will use that directory location by default. No more steps are required if you want to use this directory.

Perform the following steps to set up the log directory *logDir*, where *logDir* could be either the default log directory `/usr/spool/objy` or a nondefault log directory:

1. Log in as `root`.
2. Verify that the desired log directory exists, and, if necessary, create it. Enter:


```
mkdir logDir
```
3. Set the log directory’s permissions to enable the ODBC server (`sysctpe`) to create and update files in this directory, and to enable all users to read and write files there. Enter:


```
chmod 777 logDir
```

If you set up a nondefault *logDir*, you must enable the ODBC server to find it. To do this:

- Set the environment variable `OO_SQL_TMP_DIR` to *logDir* before you start the lock server.

Additionally, if you are configuring your workstation to restart the ODBC server upon reboot, you must arrange for the environment variable to be set before the ODBC server is restarted automatically. You can do this by editing a startup file (such as `.login`) for the *userName* account, and adding a command to set the environment variable.

NOTE The ODBC server will consult the `OO_SQL_TMP_DIR` environment variable before it looks for the default log directory.

For information on changing the log directory for ODBC log files, see Chapter 3, “Getting Started With ODBC” in *Objectivity/SQL++*.

Testing Interactive SQL++

You can test whether the Interactive SQL++ component of Objectivity/SQL++ has been set up correctly by building and running the provided sample application. This sample application builds an Objectivity/DB federated database that is then queried through Interactive SQL++.

To build and run the Interactive SQL++ sample application:

1. If you have not done so already, set up a default license file for your current Objectivity license; see “Setting Up a License File” on page 21.
2. Change your working directory to the Interactive SQL++ samples directory—for example, enter:


```
cd installDir/arch/samples/sql/ooisql
```

 You may wish to make a backup copy of this directory.
3. Edit the makefile in the Interactive SQL++ samples directory:
 - Set `INSTALL_DIR` to the location of the Objectivity/DB installation directory—for example:


```
INSTALL_DIR = /usr/object
```
 - Set `LS_HOST` to the name of the lock-server host—for example:


```
LS_HOST = myLockServerHost
```
4. Edit the demo file to set `passwd` to be the password of the Objectivity/SQL++ administrator account (`syspe`), which you created in step 2 on page 69—for example:


```
passwd=adminPassword
```
5. Check whether the lock server is running; start it, if necessary.
6. Build the sample application and create the federated database. Enter:


```
make
```
7. Run the sample application. Enter:


```
./demo
```

If Interactive SQL++ is set up correctly, you will see messages like these:

```

Creating the Objectivity/DB Database.
    (...)
Running OOISQL to create views.
    (...)
Running OOISQL to test out various SQL statements.
    (...)
Test PASSED -- The expected results were achieved.

```

NOTE If you want to repeat this demo, you must first make `clean`. However, you should not make `clean` if you plan to perform the Objectivity/SQL++ ODBC Driver demo, because the same federated database is used by that demo.

Preparing the ODBC Server for Testing

At some sites, a database administrator or a system administrator is responsible for installing Objectivity/SQL++, while individual users of ODBC-compliant client applications can install their own copies of the Objectivity/SQL++ ODBC Driver. If you are installing Objectivity/SQL++ at such a site, you probably need to set up the federated database and ODBC server so that other users can perform the Objectivity/SQL++ ODBC Driver demo.

To prepare for the Objectivity/SQL++ ODBC Driver demo, perform the following steps on the Objectivity/SQL++ ODBC server host:

1. If you have not done so already, run the entire Interactive SQL++ demo (steps 2 through 7 on page 73) to create and populate the demo federated database to be browsed. Be sure to leave the lock server running.
2. Check whether the lock server is running; start it, if necessary.
3. Check whether the ODBC server is running; start it, if necessary. For example, if you are logged in as `syspe`, enter:


```
oosqld start
```
4. If the Objectivity/SQL++ ODBC Driver demo is to be performed by another user, give that user the TCP/IP name of the ODBC server host, the service name under which the ODBC server is running (`oosqlnw`), and the location and name of the boot file (`installDir/arch/samples/sql/ooisql/DEMO`).

5. Grant all access rights to all users for the tables in the demo federated database. If you omit this step, only the Objectivity/SQL++ database administrator (`systpe`) will have access to these tables. To grant access rights to all users:
 - a. Start Interactive SQL++ for the demo federated database:

```
ooisql -user systpe -passwd adminpassword
      installDir/arch/samples/sql/ooisql/DEMO
```

where `adminpassword` is the password of the Objectivity/SQL++ administrator account (`systpe`).
 - b. Enter the `TABLE` statement to obtain a list of the demo tables:

```
table;
```
 - c. For each table listed, grant all rights to every user with a user account on the Objectivity/SQL++ ODBC server host. Enter commands such as the following:

```
grant all on tablename to public;
```
 - d. Commit the new access rights and exit from Interactive SQL++:

```
commit work;
exit
```

Users can now perform the Objectivity/SQL++ ODBC Driver demo described in the Objectivity/SQL++ ODBC Driver installation chapter of *Installation and Platform Notes for Windows, Installation and Platform Notes for UNIX* (this book), or *Objectivity/SQL++ ODBC Driver User's Guide*.

The demo can be repeated as long as the lock server and the Objectivity/SQL++ ODBC server are both running, and the demo federated database exists.

Objectivity/SQL++ ODBC Driver Installation

This chapter describes the requirements and steps for installing the Objectivity/SQL++ ODBC Driver (Objectivity/ODBC) on a UNIX platform. Objectivity/ODBC enables ODBC-compliant client applications, such as StarOffice, to access Objectivity/DB federated databases through an Objectivity/SQL++ ODBC server.

Objectivity/ODBC supports ODBC 3.0.

See Chapter 8, “Objectivity/SQL++ Installation,” for information about the Objectivity/SQL++ ODBC server.

System Requirements

You can install Objectivity/ODBC on any of the architectures listed in Table 1-1 on page 14.

Software Requirements

Objectivity/ODBC requires that an Objectivity/SQL++ ODBC server be installed in the same network. The Objectivity/SQL++ ODBC server must implement ODBC 3.0.

A C++ development environment is required for building and running the sample application that verifies Objectivity/ODBC installation.

Objectivity/ODBC requires that a third-party ODBC driver-manager be installed on each UNIX workstation that is to run an ODBC-compliant client application. An ODBC driver-manager is the UNIX counterpart of the Microsoft ODBC Administrator. For convenience, you can use the ODBC driver-manager that is distributed with Objectivity/ODBC—namely, unixODBC DriverManager. This driver-manager is installed automatically when you install Objectivity/ODBC.

As distributed, unixODBC DriverManager provides a text-based user interface for registering the driver and adding data sources. If you prefer a graphical user

interface, you can optionally obtain the entire unixODBC DriverManager, along with its dependencies and manuals, from the unixODBC Project Web site at www.unixodbc.org.

NOTE The steps in this chapter assume you are using the text-based unixODBC DriverManager as it is distributed and installed with Objectivity/ODBC.

Installing Objectivity/ODBC

To install Objectivity/ODBC:

1. Log in to your workstation as `root`.
2. Verify that required software has been completely and correctly installed (see “Software Requirements” on page 77).
3. If you have not done so already, obtain the Objectivity installation files—for example, by downloading them from the Objectivity Technical Support Web site.
4. Run the Objectivity installation script with a command such as the following:

```
./install.sh
```
5. When prompted, select **Custom Installation**; then select Objectivity/SQL++ ODBC Driver by entering its item number from the displayed list.

NOTE You must have a license for every product you install.

6. At the directory prompt, specify the directory (*installDir*) in which to install Objectivity/ODBC. (If Objectivity/DB is also installed on your machine, the Objectivity/DB installation directory is recommended—for example, `/usr/object`.)

The installation script:

- Places the release files in subdirectories of *installDir/arch*, where *arch* is the architecture name for your platform.
- Runs the installation verification test `ooverify`. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 31.

Preparing to Use Objectivity/ODBC

After the installation script completes, perform the following steps:

1. Familiarize yourself with your installation by reading “Release Files for Objectivity/ODBC” on page 79.
2. Set the environment variable `OO_SQL_DIR` to the fully qualified pathname of the Objectivity/DB installation directory (*installDir/arch*). You may wish to set the variable in a startup file such as `.login`.

NOTE You must set the `OO_SQL_DIR` environment variable for every user that will interact with the provided ODBC driver-manager—for example, to list registered drivers or to add data sources. This variable enables the ODBC driver-manager to use *installDir/arch/etc/sql* as its system directory.

3. Verify that Objectivity/ODBC has been registered with the provided ODBC driver-manager. Follow the steps in “Checking and Registering the Driver” on page 80.
4. Add the desired Objectivity/DB federated databases as data sources. Follow the steps in “Adding Objectivity/DB Data Sources” on page 81.
5. Configure TCP/IP to enable Objectivity/ODBC to communicate with an Objectivity/SQL++ ODBC server. Follow the steps in “Configuring TCP/IP” on page 83.
6. (Optional) Test Objectivity/ODBC with an Objectivity/SQL++ ODBC server by browsing the provided demo federated database (see “Testing Objectivity/ODBC” on page 84).

Release Files for Objectivity/ODBC

When you install Objectivity/ODBC, its files are organized in subdirectories of the *installDir/arch* directory, as shown in Table 9-1.

Table 9-1: Release Files in *installDir/arch*

Subdirectory	Contains
bin	Tool for adding data sources and registering Objectivity/ODBC with unixODBC DriverManager
samples/sql/odbc	Application for demonstration and testing
doc	PDF file for the online book <i>Objectivity/SQL++ ODBC Driver User's Guide</i>

Table 9-1: Release Files in *installDir/arch* (Continued)

Subdirectory	Contains
etc/sql	Template files used for adding data sources and registering Objectivity/ODBC with unixODBC DriverManager
include	Include files for developing custom ODBC-compliant applications
lib	Objectivity/ODBC shared library (the driver) and unixODBC shared libraries (the driver-manager)

Checking and Registering the Driver

When you install Objectivity/ODBC, it is automatically *registered* with the provided ODBC driver-manager (unixODBC DriverManager). Objectivity/ODBC must be registered so that unixODBC DriverManager can load it whenever an ODBC-compliant client application requests data from an Objectivity/DB federated database.

After the Objectivity installation script completes, you should query the provided ODBC driver-manager to check whether Objectivity/ODBC was registered correctly.

Checking Whether Objectivity/ODBC is Registered

To verify that Objectivity/ODBC is registered with unixODBC DriverManager:

1. Enter the following at a command prompt:

```
odbcinst -q -d
```
2. Look for the following driver name in the list of registered ODBC drivers:

```
ObjectivitySQL
```
3. If you do not see `ObjectivitySQL` listed as a registered ODBC driver:
 - a. Make sure the `OO_SQL_DIR` environment variable for your account is set to the fully qualified pathname of the Objectivity/DB installation directory (*installDir/arch*).
 - b. Repeat step 1.

Registering Objectivity/ODBC

If Objectivity/ODBC is not registered with unixODBC DriverManager or if you want to change some aspect of its registration, you can perform the registration steps yourself:

1. Make sure the `OO_SQL_DIR` environment variable for your account is set to the fully qualified pathname of the Objectivity/DB installation directory (`installDir/arch`).
2. Change your working directory to the Objectivity/SQL++ support directory.
Enter:

```
cd installDir/arch/etc/sql
```
3. Inspect the driver template file `odbcinst.ini_template`. If necessary, edit the file in a text editor—for example, to enter the correct pathname for the Objectivity/DB installation directory.
4. Register Objectivity/ODBC using the information in the driver template file.
Enter:

```
odbcinst -i -d -f odbcinst.ini_template
```
5. Check for the resulting driver file:

```
installDir/arch/etc/sql/odbcinst.ini
```

NOTE To get help for the `odbcinst` tool, run it with no options or arguments.

Adding Objectivity/DB Data Sources

You enable ODBC-compliant client applications to access an Objectivity/DB federated database by adding a *data source* for it. In general, a data source identifies the data to be accessed and the means of accessing it (for example, host and network information). The data source you add for a federated database specifies its boot file and a host running a particular Objectivity/SQL++ ODBC server.

You can add a *personal data source*, which is specific to applications running under your user account, or a *system data source*, which is available to applications running under any user account.

Before you add a data source for a federated database:

- Identify the Objectivity/SQL++ ODBC server that will access the federated database, and obtain:
 - The TCP/IP name of the host that runs the ODBC server.
 - The service name under which the ODBC server is running.

- Obtain the location and name of the federated database's boot file.
- Ensure that you have a valid user account on the host running the Objectivity/SQL++ ODBC server.

To add a data source for a federated database:

1. Log in under an appropriate user account. In particular, you must log in under your user account if you are adding a personal data source.
2. Make sure the `OO_SQL_DIR` environment variable for your account is set to the fully qualified pathname of the Objectivity/DB installation directory (*installDir/arch*).
3. Copy the provided data-source template file to a separate location and change your working directory to this new location. For example, enter the following, where *homeDir* represents your home directory:


```
cp installDir/arch/etc/sql/odbc.ini_template
   homeDir/odbc.ini_tmpl
cd
```
4. Open your copy of the data-source template file in a text editor.
5. Edit the first line so that it contains the desired data-source name enclosed in brackets—for example, `[MyObjyData]`. The data-source name should uniquely identify the data source.
6. Edit the subsequent lines so that each keyword has an appropriate value. Each keyword-value entry should consist of a single line:

Description	=	A description for the data source.
Driver	=	The name under which Objectivity/ODBC is registered with the ODBC driver-manager—for example, ObjectivitySQL.
Host	=	The TCP/IP name of the host machine running the Objectivity/SQL++ ODBC server (the <code>oosqld</code> process).
Database	=	Fully qualified pathname of the federated database's boot file on the specified host.
User ID	=	The username of your Objectivity/SQL++ account. This is normally your login account on the ODBC server host, provided that this account has been granted access rights to tables in the federated database (see your Objectivity/SQL++ database administrator). The Objectivity/SQL++ database administrator account (<code>sysupe</code>) has access to all tables.
Password	=	Password for the account you entered in the User ID field. The password you enter is <i>not</i> encoded before it is sent across the network.
Service	=	The service name of the ODBC server on its host (<code>oosqlnw</code>).

NOTE The Host, Database, and Service fields together must not exceed a string-length of 988 characters.

7. Save the data-source template file.
8. Add the data source using the information in the template file you edited.

To add a personal data source, enter:

```
odbcinst -i -s -f odbc.ini_template
```

To add a system data source, enter:

```
odbcinst -i -s -f odbc.ini_template -l
```

9. Verify that the data source was successfully added. Enter:

```
odbcinst -q -s
```

Adding a personal data source creates a file called `.odbc.ini` in your home directory. Adding a system data source creates a file called `installDir/arch/etc/sql/odbc.ini`.

Configuring TCP/IP

You or your system administrator must perform the following steps to enable your ODBC-compliant application to communicate with the Objectivity/SQL++ ODBC server across the network.

Identifying the ODBC Server's Host to TCP/IP

TCP/IP must be able to recognize the hostname you specify when you register a data source. That is, TCP/IP must be able to convert the hostname into an Internet address. Many sites use the TCP/IP `hosts` file to map hostnames to Internet addresses, although some sites use domain name servers for this purpose.

You should verify that the computer on which you installed Objectivity/ODBC recognizes a valid hostname for the computer on which the ODBC server is running. For example, to verify that your computer recognizes `hostname`:

- Enter the following command at a command prompt on your computer:

```
ping hostname
```

Specifying the ODBC Server's Port Number

The Objectivity/SQL++ ODBC Driver and the Objectivity/SQL++ ODBC server must be able to communicate through the same TCP/IP port. Consequently, you

must register the port number with TCP/IP on each Objectivity/ODBC (driver) host. To do so:

1. Find the TCP port assigned to the `oosqlnw` service on the host running the Objectivity/SQL++ ODBC server. The port is normally 1990.
2. On the host where you installed Objectivity/ODBC (the driver), open the TCP/IP `services` file (typically, `/etc/services`).
3. Add the following entry to the TCP/IP `services` file:

```
oosqlnw portNumber/tcp      # Objectivity/SQL++ server
```

 where `portNumber` is the TCP port number you found in step 1—for example:

```
oosqlnw      1990/tcp          # Objectivity/SQL++ server
```
4. Send a hang-up signal to your `inetd` process. For example, if `inetdPid` is the process identifier for the `inetd` process, enter:

```
kill -HUP inetdPid
```

Note: You must assign the *same* port to the `oosqlnw` service on every Objectivity/SQL++ ODBC Driver host.

Testing Objectivity/ODBC

You can verify the correct operation of Objectivity/ODBC in combination with an Objectivity/SQL++ ODBC server by running the following test. This test compiles and links a sample ODBC application and then runs it to access a demonstration federated database. You need Objectivity/C++ to run this test.

The demo federated database is provided with Objectivity/SQL++ on the ODBC server host.

The sample application is a C++ application that calls ODBC 3.0 functions to query and modify a federated database. You can inspect the sample to see how such an application is compiled and linked.

To build and run the sample ODBC-compliant application:

1. Verify that the demo federated database and the Objectivity/SQL++ ODBC server have been set up for this test:
 - If your ODBC server runs on Windows, see “Preparing the ODBC Server for Testing” in Chapter 9 in *Installation and Platform Notes for Windows*.
 - If your ODBC server runs on UNIX, see “Preparing the ODBC Server for Testing” in Chapter 8 in this book.

The Objectivity/SQL++ ODBC server must be running.

2. On the host where you installed Objectivity/ODBC (the driver), change your working directory to the Objectivity/SQL++ support directory. Enter:

```
cd installDir/arch/etc/sql
```

3. In the Objectivity/SQL++ support directory, open the data-source template file `odbc.ini_template` in a text editor.

- a. Make sure the data-source name on the first line is `[DEMO]`.
- b. Update each entry as necessary to reflect your environment:

```
Description = DEMO DSN for Objectivity SQL++ ODBC demo
Driver       = driverName (probably ObjectivitySQL)
Host        = hostName (name of the host running your Objectivity/SQL++
              ODBC server)
Database    = Fully qualified boot-file path for the demo federated database
              under the Objectivity/DB installation directory (installDir) on
              hostName—that is:
              installDir\samples\sql\ooisql\DEMO    (Windows)
              installDir/arch/samples/sql/ooisql/DEMO (UNIX)
User ID     = A username that has been granted access rights for the tables in
              the demo federated database. By default, this is the
              Objectivity/SQL++ database administrator account (sysstpe).
Password    = Password for the account you entered in the User ID field.
Service     = oosqlnw
```

4. Save the data-source template file.
5. Add the `[DEMO]` data source as a personal data source. Enter:

```
odbcinst -i -s -f odbc.ini_template
```

For details, see “Adding Objectivity/DB Data Sources” on page 81.

6. Change your working directory to the Objectivity/ODBC samples directory. In a command window, enter:

```
cd installDir/arch/samples/sql/odbc
```

You may wish to make a backup copy of this directory.

7. Edit `makefile` in the Objectivity/ODBC samples directory.
 - Set `INSTALL_DIR` to be the location of the Objectivity/DB installation directory—for example:


```
INSTALL_DIR = /usr/object
```
 - Set `LS_HOST` to be the name of the host running the lock server—for example:


```
LS_HOST = myLockServerHost
```

8. Check whether the lock server is running; start it, if necessary.
9. Build and run the sample application. At the command prompt, enter:
make
10. Run the sample application. Enter:
./demo.sh
Press Return at each prompt to continue the demo.

If the Objectivity/SQL++ ODBC programming interface is set up correctly, you will see messages like these:

```
Welcome to the Objectivity/SQL++ ODBC demo
...
Building the ooODBCdemo executable
...
Running the demo...
Demo complete.
Comparing the results.
Test PASSED -- The expected results were achieved.
No errors.
```

Objectivity/DB High Availability Installation

This chapter describes the requirements and steps for installing Objectivity/DB High Availability (Objectivity/HA) on a UNIX platform. Objectivity/HA improves the availability of distributed data in an Objectivity/DB federated database. With Objectivity/HA, you can:

- Organize the data files of an Objectivity/DB federated database into independently managed groups called *autonomous partitions*. When a network or machine failure prevents access to data in one partition, users can continue to access the data controlled by the remaining partitions.
This aspect of Objectivity/HA is sometimes called *fault tolerant option*.
- Create and distribute managed copies of each database (called *database images*) among multiple autonomous partitions. When a failure prevents access to one copy, you can continue to access copies in other partitions.
This aspect of Objectivity/HA is sometimes called *data replication option*.

System Requirements

You can install Objectivity/HA on any of the UNIX architectures listed in Table 1-1 on page 14.

Software Requirements

Objectivity/HA requires that the following software be installed on your system:

- Objectivity/DB (see Chapter 1)
The Advanced Multithreaded Server (AMS) must be installed and configured on every host that is to store an autonomous partition or a replicated database (see “Setting Up the Advanced Multithreaded Server” on page 27).

Installing Objectivity/HA

To install Objectivity/HA:

1. Log in to your workstation as `root`.
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” above).
3. If you have not done so already, obtain the Objectivity installation files as described in “Accessing Objectivity Installation Files” on page 15.
4. Run the Objectivity installation script with a command such as the following:

```
./install.sh
```
5. When prompted, select **Custom Installation**; then select Objectivity/HA from the displayed list.

NOTE You must be licensed for every product you install.

6. At the directory prompt, specify the directory in which Objectivity/DB is installed (*installDir*). If you get an error message reporting insufficient disk space or incorrect permissions, press Enter to exit the installation script; then fix the problem and restart the installation script (see step 4).

The installation script:

- Places the release files in subdirectories of *installDir/arch*, where *arch* is the architecture name for your platform.
- Runs the installation verification test `ooverify`. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 31.

Preparing to Use Objectivity/HA

After the installation script completes, perform the following steps:

1. Familiarize yourself with your installation by reading “Release Files for Objectivity/HA” on page 89.
2. Verify that AMS is installed and configured on every host that is to store an autonomous partition or a replicated database (see “Setting Up the Advanced Multithreaded Server” on page 27). Although AMS need not be running when you create an original database image, you must start AMS before you can create additional database images.
3. Read:
 - *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the file `installDir/arch/doc/ReleaseNotes.pdf`.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

Release Files for Objectivity/HA

When you install Objectivity/HA, its files are organized in subdirectories of the `installDir/arch` directory, as shown in Table 10-1.

Table 10-1: Objectivity/HA Release Files in `installDir/arch`

Subdirectory	Contains
bin	Executables for Objectivity/HA tools
doc	PDF file for the online book <i>Objectivity/DB High Availability</i>

Objectivity/DB Parallel Query Engine Installation

This chapter describes the requirements and steps for installing Objectivity/DB Parallel Query Engine (Objectivity/PQE) on a UNIX platform. Objectivity/PQE enables an application to use *parallel queries* to find qualified objects of a particular class, where a parallel query consists of subtasks which are executed in parallel by a *query server* provided with Objectivity/PQE.

System Requirements

You can install Objectivity/PQE on any of the UNIX architectures listed in Table 1-1 on page 14.

Software Requirements

Objectivity/PQE requires that the following software be installed on your system:

- Objectivity/DB (see Chapter 1)

Installing Objectivity/PQE

To install Objectivity/PQE:

1. Log in to your workstation as `root`.
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” above).
3. If you have not done so already, obtain the Objectivity installation files as described in “Accessing Objectivity Installation Files” on page 15.
4. Run the Objectivity installation script with a command such as the following:

```
./install.sh
```

5. When prompted, select **Custom Installation**; then select Objectivity/PQE from the displayed list.

NOTE You must be licensed for every product you install.

6. At the directory prompt, specify the directory in which Objectivity/DB is installed (*installDir*). If you get an error message reporting insufficient disk space or incorrect permissions, press Enter to exit the installation script; then fix the problem and restart the installation script (see step 4).

The installation script:

- Places the release files in subdirectories of *installDir/arch*, where *arch* is the architecture name for your platform.
- Runs the installation verification test *ooverify*. If *ooverify* displays error messages, see “Troubleshooting Installation” on page 31.

Preparing to Use Objectivity/PQE

After the installation script completes, perform the following steps:

1. Familiarize yourself with your installation by reading “Release Files for Objectivity/PQE” on page 93.
2. Set up the query server by following the steps in “Setting Up the Query Server” on page 93.
3. Read:
 - *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the file *installDir/arch/doc/ReleaseNotes.pdf*.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

Release Files for Objectivity/PQE

When you install Objectivity/PQE, its files are organized in subdirectories of the *installDir/arch* directory, as shown in Table 11-1.

Table 11-1: Objectivity/PQE Release Files in *installDir/arch*

Subdirectory	Contains
bin	Executables for Objectivity/PQE query server
include	Include file <code>ooPQE.h</code> for customizing Objectivity/PQE
lib	Shared libraries for Objectivity/PQE

Setting Up the Query Server

Objectivity/PQE uses a query server to execute the subtasks of a query in parallel.

You can run the query server on one or more hosts. If you want to run a query locally on a workstation that does not contain the Objectivity/DB installation, you can copy the `ooqueryserver` and `ooqs` executables to that workstation, along with the shared libraries they reference (use a command such as `ldd` to identify the required libraries).

For security purposes, you should consider running the query server under a special-purpose user account in a group that can be granted just the minimum necessary permissions.

You may start a query server at any time after installation, but you must start it before running any database applications that require parallel queries. It is common practice to configure your workstation so that the query server starts whenever the machine reboots.

Starting the Query Server

Perform the following steps when you start the query server on a particular workstation for the first time:

1. Log in as `root`.
2. Start the query server to verify that it is installed properly. To start the query server under the user account *username*, you can enter a command such as the following:

```
su - username -c "installDir/arch/bin/ooqueryserver -start"
```

3. If the query server does not start, check whether a port conflict exists; see “Specifying the Query Server’s Port Number” on page 94.
4. (Optional) Configure your workstation to start the query server whenever the machine reboots. To do this, edit the workstation’s startup script (usually `/etc/rc.local` or `/etc/init.d`) and add a command such as the one shown in step 2.

Checking the Query Server

Perform the following step to check that the query server is running:

- Enter the following command:

```
installDir/arch/bin/ooqueryserver -check
```

Specifying the Query Server’s Port Number

The query server will not start if another service already uses the TCP/IP port that the lock server expected to use. If you cannot reassign the other services to a different port, you may have to change the default port for the lock server. To do this, see Chapter 10, “Using a Query Server,” in *Objectivity/DB Administration*.

Using Objectivity/PQE

After Objectivity/PQE is installed, you can initiate a parallel query from within an Objectivity/DB database application. You accomplish this by adding the appropriate function call to the application, as described in the documentation for your Objectivity/DB programming interface.

An application that starts a parallel query must be linked with the shared Objectivity/DB library, not the static library (see “Linking Applications to Objectivity/DB” on page 99). No extra steps are required to compile and link an application that uses Objectivity/PQE. The Objectivity/PQE shared library is loaded automatically at runtime when the application initiates a parallel query.

Objectivity/IPLS Installation

This chapter describes the requirements and steps for installing Objectivity/DB In-Process Lock Server (Objectivity/IPLS) on a UNIX platform. Objectivity/IPLS enables you to run a lock server as part of a C++, Java, or Smalltalk database application instead of running the lock server as a separate process.

System Requirements

You can install Objectivity/IPLS on any of the UNIX architectures listed in Table 1-1 on page 14.

Software Requirements

Objectivity/IPLS requires that the following software be installed on your system:

- Objectivity/DB (see Chapter 1)

Installing Objectivity/IPLS

To install Objectivity/IPLS:

1. Log in to your workstation as `root`.
2. Verify that all required software has been completely and correctly installed.
3. If you have not done so already, obtain the Objectivity installation files as described in “Accessing Objectivity Installation Files” on page 15.
4. Run the Objectivity installation script with a command such as the following:

```
./install.sh
```
5. When prompted, select **Custom Installation**; then select Objectivity/IPLS from the displayed list.

NOTE You must be licensed for every product you install.

6. At the directory prompt, specify the directory in which Objectivity/DB is installed (*installDir*). If you get an error message reporting insufficient disk space or incorrect permissions, press Enter to exit the installation script; then fix the problem and restart the installation script (see step 4).

The installation script:

- Places the release files in subdirectories of *installDir/arch*, where *arch* is the architecture name for your platform.
- Runs the installation verification test `ooverify`. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 31.

Preparing to Use Objectivity/IPLS

After the installation script completes, perform the following steps:

1. Familiarize yourself with your installation by reading “Release Files for Objectivity/IPLS” on page 96.
2. If necessary, set up a log directory in which the in-process lock server can create log files. Follow the steps in “Setting Up a Log Directory for the Lock Server” in Chapter 1.
3. Read:
 - “Using Objectivity/IPLS” on page 97
 - *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the file *installDir/arch/doc/ReleaseNotes.pdf*.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

Release Files for Objectivity/IPLS

When you install Objectivity/IPLS, its file is placed in a subdirectory of the *installDir/arch* directory, as shown in Table 12-1.

Table 12-1: Objectivity/IPLS Release Files in *installDir/arch*

Subdirectory	Description
lib	Shared library for Objectivity/IPLS

Using Objectivity/IPLS

After Objectivity/IPLS is installed, you can start an in-process lock server from within a C++, Java, or Smalltalk database application. You accomplish this by adding an appropriate function call to the application, as described in the chapter about Objectivity/IPLS in *Objectivity/C++ Programmer's Guide*, *Objectivity for Java Programmer's Guide*, or *Objectivity/Smalltalk for VisualWorks*.

An application that starts an in-process lock server must be linked with the shared Objectivity/DB library, not the static library (see "Linking Applications to Objectivity/DB" on page 99). No extra steps are required to compile and link an IPLS application. The Objectivity/IPLS shared library is loaded automatically at runtime when the application starts the in-process lock server.

When an in-process lock server is started, the application that starts it becomes the lock server for the workstation on which it is running, and you must stop any other lock-server process that is running on the same workstation. For information about managing in-process and standard lock servers, see Chapter 8, "Using a Lock Server," in *Objectivity/DB Administration*.

A

C++ Application Development

This appendix gives platform-specific details about developing Objectivity/C++ applications on a UNIX platform. You should use this appendix in conjunction with Objectivity/C++ and Objectivity/DDL books.

This appendix provides information about:

- Linking applications to Objectivity/DB
- Linking applications to additional Objectivity products and features
- Using makefiles
- Application programming issues and debugging

See also Appendix C, “Sample Applications for C++ and Java.”

Linking Applications to Objectivity/DB

You can link C++ database applications to Objectivity/DB static or shared libraries. Objectivity/C++ is compatible with ANSI C++.

Libraries for Static Linking

Table A-1 shows Objectivity/DB static runtime libraries for C++ applications.

Table A-1: Objectivity/DB Static Link Libraries

Library File	Description
liboo.a	Objectivity/DB standard library. ^a
liboo_adm.a	Objectivity/DB administration library. Add this library to your link line <i>before</i> liboo.a if you are using Objectivity/C++ member functions to perform recovery operations.
liboo_dbx.a	Debug version of the Objectivity/DB standard library. Add this library to your link line <i>instead of</i> liboo.a if you want to use debug mode or Objectivity/DB debugger tools (see page 108). This library performs more runtime checking than liboo.a.

a. If your application uses predicate scans, you must link to the Objectivity/DB shared library (not liboo.a).

Static Link Rules

Table A-2 summarizes link rules for linking with the standard Objectivity/DB static runtime library.

This table uses the following convention:

installDir Objectivity/DB installation directory—by default, /usr/object

Table A-2: Static Link Rules

Architecture	Link Rule
hprisc	-AA -mt -L <i>installDir</i> /hprisc/lib -looo -lnsl -lpthread -ldl
hpuxia64	<i>installDir</i> /hpuxia64/lib/liboo.a -AA +DD64 -mt -lpthread -lnsl
linux86_64	<i>installDir</i> /linux86_64/lib/liboo.a -pthread -ldl
linux86gcc3	<i>installDir</i> /linux86gcc3/lib/liboo.a -lrpcsvc -lnsl -pthread -ldl
solaris7	-L <i>installDir</i> /solaris7/lib -Bstatic -looo -Bdynamic -lpthread -lrpcsvc -lnsl -lsocket -ldl -mt

Table A-2: Static Link Rules (Continued)

solaris86_64	-LinstallDir/solaris86_64/lib -Bstatic -lo -Bdynamic -lpthread -lrpcsvc -lnsl -lsocket -ldl -xarch=amd64 -mt
sparc64 ^a	-LinstallDir/sparc64/lib -Bstatic -lo -Bdynamic -lpthread -lrpcsvc -lnsl -lsocket -ldl -m64 -mt

- a. If you are using version 5.7 or earlier of the Sun C++ compiler, you will need to specify the `-xarch=v9` flag instead of the `-m64` flag.

Linking to Shared Libraries

Objectivity/DB provides shared library versions of `liboo.a`. The names of these shared libraries vary on different architectures and contain digits `x.x` corresponding to the current Objectivity/DB release. To link Objectivity/DB applications to shared libraries on UNIX, choose a link rule depending on:

- Whether you are linking applications for development or end-user deployment.
- The architecture you are using—for example, `solaris7`. See Table 1-1 on page 14 for more information about architecture names.

If your application uses persistent collections or the C++ programming interface of other Objectivity products, you must add other libraries to your link line (see “Linking to Additional Objectivity Products and Features” on page 104).

Link Rules for Development Environments

Table A-3 summarizes C++ UNIX shared library link rules for development environments. This table uses the following conventions:

`installDir` Objectivity/DB installation directory—by default, `/usr/object`

`x.x` Digits corresponding to the current Objectivity/DB release

Table A-3: Shared Library Link Rules for Development

Architecture	Link Rule
hprisc	-AA -mt -LinstallDir/hprisc/lib -loox.x -lpthread
hpuxia64	-LinstallDir/lib/hpuxia64 -loox.x -AA +DD64 -mt -lpthread -lnsl

Table A-3: Shared Library Link Rules for Development (Continued)

linux86_64	-LinstallDir/linux86_64/lib -Bdynamic -lso -Xlinker -rpath-link=installDir/linux86_64/lib -pthread -ldl
linux86gcc3	-LinstallDir/linux86gcc3/lib -Bdynamic -lso -lrpcsvc -lnsl -ldl -pthread
solaris7	-LinstallDir/solaris7/lib -lso.x.x -mt
solaris86_64	-LinstallDir/solaris86_64/lib -lso.x.x -mt -xarch=amd64
sparc64 ^a	-LinstallDir/sparc64/lib -lso.x.x -mt -m64

- a. If you are using version 5.7 or earlier of the Sun C++ compiler, you will need to specify the `-xarch=v9` flag instead of the `-m64` flag.

Link Rules for End-User Environments

Table A-4 provides information for embedding the runtime shared-library search path so that you will be able to find Objectivity/DB libraries in the Installation directory at an end-user site. The options in this table should be used *in addition* to the options listed in Table A-3 when linking applications for deployment.

Table A-4 uses the following conventions:

installDir Objectivity/DB installation directory—by default, /usr/object
endUserInstallDir Installation directory at an end-user site
x.x Digits corresponding to the current Objectivity/DB release

Table A-4: Shared Library Link Rules for Deployment

Architecture	Link Rule
hprisc	-Wl,+s -Wl,+b <i>endUserInstallDir</i> See the ld(1) man page for more information.
hpuxia64	-Wl,+s -Wl,+b <i>endUserInstallDir</i>
linux86_64	-Xlinker -rpath-link= <i>installDir</i> /linux86_64/lib -Xlinker -rpath= <i>endUserInstallDir</i>
linux86gcc3	-Xlinker -rpath-link= <i>installDir</i> /linux86gcc3/lib -Xlinker -rpath= <i>endUserInstallDir</i>
solaris7	Define LD_RUN_PATH prior to compiling and linking.
solaris86_64	Define LD_RUN_PATH prior to compiling and linking.
sparc64	Define LD_RUN_PATH prior to compiling and linking.

Linking With Purify

If you are linking your application with Purify, you may want to suppress Objectivity/DB UMRs when you run your application. These UMRs are produced in error and can safely be ignored. To do so, add the following entries to your .purify file:

```
suppress umr write; ...; onmWrite;
suppress umr write; ...; onmSeekWrite;
```


If Your Application Uses Persistent Collections

If the persistent collections used by your application are all implemented with the Release 9.3 (or later) scalable collection structure (for example, `ooTreeSetX`), you should include the header file `ooCollectionBase.h`. No special linking is necessary, other than linking to the Objectivity/DB library.

If any of the collections used by your application are implemented with the pre-Release 9.3 scalable collection structure (for example, `ooTreeSet`), you must include the header file `ooCollections.h`. Table A-6 lists the collection-specific shared libraries:

Table A-6: Shared Libraries For pre-Release 9.3 Persistent Collections

To Use	Link to
Objectivity/C++ persistent collections	-l <code>oo_co</code> On the <code>linux*</code> architectures
	-l <code>oo_co.x.x</code> On all other architectures
	Link to the Objectivity/DB shared library (not <code>liboo.a</code>).

Linking a Lock-Server Performance-Monitoring Program

Table A-7 lists the libraries for linking a custom C++ program that monitors how your database applications interact with a running lock server. The information collected by such a program can help you analyze application performance (for more information, see the *Monitoring Lock-Server Performance* online book).

You can link the program to either the static or shared library in Table A-7. If you link to the static library, you must also link to `liboo.a`. Otherwise, if you link to the shared library, you must also link to the Objectivity/DB shared library using the appropriate link rule given in Table A-3.

Table A-7: Linking a Lock-Server Performance-Monitoring Program

Link to	Description
<code>liboolspm.a</code>	Static runtime library
-l <code>oolspm</code> -l <code>oolspm.x.x</code>	Shared library on the <code>linux*</code> architectures Shared library on all other architectures

Using Makefiles

You can use a makefile to run the DDL processor and then compile and link your application with the DDL-generated header and implementation files. You can use the makefile from any C++ sample application as a template for your own makefile (see “Testing Objectivity/C++ Setup” on page 38). The sample makefiles are in:

```
installDir/arch/samples/cxx/appDir/Makefile
```

where *appDir* is the name of a directory corresponding to a sample application—for example, *helloWorld*.

As you inspect the sample makefiles, keep in mind that every sample makefile *appDir/Makefile*:

- Obtains configuration information from the file *installDir/arch/samples/cxx/include.mk*.
- Has dependencies on subordinate makefiles *appDir/ddlFiles/Makefile* and *appDir/src/Makefile*, in subdirectories.

Compiler Flags

In general, you should use the same compiler flags as those used in the sample makefiles. Note:

- On the *hprisc* and *hpuxia64* architectures, you must use the *-AA* and *-mt* compiler flags.
- On the *hpuxia64* architecture, you must use the *+DD64* compiler flag.
- On the *linux86_64* and *linux86gcc3* architectures, you should avoid using the *-Weffc++* and *-Wnon-virtual-dtor* options, because they will result in compiler warnings that cannot be fixed. (Alternatively, you can suppress these warnings by accessing the Objectivity/DB include directory with the *-isystem* option.)

Application Programming Issues

Signal Handling

The Objectivity/DB predefined signal handler catches the following UNIX signals: SIGINT, SIGQUIT, SIGILL, SIGABRT, SIGFPE, SIGBUS, SIGSEGV, SIGHUP, SIGTERM, SIGEMT, and SIGTRAP.

Stack Size for Multithreaded Applications

In a multithreaded application, you may need to increase the stack size for each thread that executes Objectivity/DB operations. This is because the default thread stack size on some platforms may be insufficient to accommodate an Objectivity context. A minimum of 1 megabyte is recommended.

File Descriptor Limit

When running multithreaded programs with large numbers of threads, your process may reach the file descriptor limit. When this limit is reached, Objectivity/C++ operations may fail because the process cannot obtain a file descriptor.

To eliminate such errors, you should increase the file descriptor limit. The `cs`h and `ks`h commands for increasing the limit are shown in the following example. If you still experience errors, you should increase the limit incrementally.

EXAMPLE**cs**h:

```
limit descriptors 256
```

ksh:

```
ulimit -n 256
```

Debugging an Application

While debugging an Objectivity/C++ application, you can:

- Use Objectivity/DB tools for inspecting and changing federated databases (see Chapter 5, “Debugging a Federated Database,” in *Objectivity/DB Administration*).
- Run your application in debug mode for data verification and event tracing (see *Objectivity/C++ Programmer’s Guide*).

In either case, you must first prepare your application for debugging, as described in the following subsection. The remainder of this section describes how to print handles and persistent objects from the dbx debugger.

Preparing to Debug an Application

Before you can debug your Objectivity/C++ application, you must recompile your source code with the debug flag for your compiler (for example, `-g`), and relink your application to the Objectivity/DB library `liboo_dbx.a` instead of `liboo.a`.

Printing Handles

While using dbx, you can print a variable whose value is a handle. Doing so displays the object identifier of the persistent object that the handle references.

EXAMPLE Assume your application sets a handle variable `oopvar` to reference a persistent object whose object identifier is 2-2-25-144. To print the variable `oopvar`, you enter the following dbx debugger command:

```
print oopvar
```

The variable `oopvar` is printed as follows:

```
oopvar = {  
  _DB    = 2  
  _OC    = 2  
  _page  = 25  
  _slot  = 144  
}
```

Printing Objects

While using the dbx debugger, you can print the contents of a persistent object from the object's handle. The easiest way to do this is to use the `ooprint` convenience function. See Chapter 5, "Debugging a Federated Database," in *Objectivity/DB Administration*.

B

Java Application Development

This appendix gives platform-specific details about developing Objectivity for Java applications on a UNIX platform. You should use this appendix in conjunction with the Objectivity for Java programmer's guide.

This appendix provides information about:

- Application programming issues
- Java command line options

Application Programming Issues

Signal Handling

The Objectivity/DB predefined signal handler catches the following UNIX signals: SIGINT, SIGQUIT, SIGILL, SIGABRT, SIGFPE, SIGBUS, SIGSEGV, SIGHUP, SIGTERM, SIGEMT, and SIGTRAP.

By default, Objectivity for Java does not install the Objectivity/DB signal handler because some Java runtime environments do not support chaining of signal handlers. In such cases, installing the Objectivity/DB signal handler results in a segmentation violation rather than an orderly Java VM shutdown.

On the `hprisc`, `hpuxia64`, `linux86_64`, `linux86gcc3`, `solaris7`, and `sparc64` architectures, you can install the Objectivity/DB signal handler if you use a JDK which supports signal-chaining. To use this functionality, you must set the `LD_PRELOAD` environment variable as follows:

- On `linux86gcc3`, `solaris7`, and `sparc64`, set `LD_PRELOAD` to `<libjvm.so dir>/libjsig.so`. This shared library must be loaded before `libc/libthread/libpthread`.
- On `hprisc`, set `LD_PRELOAD` to `<libjvm.sl dir>/libjsig.sl`. This shared library must be loaded before `libc`.
- On `hpuxia64`, set `LD_PRELOAD` to `<libjvm.so dir>/libjsig.so`. This shared library must be loaded before `libc`.

For more information, see the vendor's release notes for the JDK.

As an alternative to installing the Objectivity/DB signal handler, you can turn off some or all signals. See the section on the Objectivity/DB signal handler in Chapter 4, "Controlling Interaction With Objectivity/DB," of the *Objectivity for Java Programmer's Guide*.

File-Descriptor Limit

When running multithreaded programs with large numbers of threads, your process may reach the file-descriptor limit. When this limit is reached, the Java virtual machine unloads classes, and, at a later time, may report that it cannot find the definition of a class. Alternatively, Objectivity for Java operations may fail, with the cause being the inability to obtain a file descriptor.

To eliminate these errors, you should increase the file-descriptor limit. The `cs`h and `ks`h commands to increase the limit are shown below. If you still experience errors, you should increase the limit incrementally.

EXAMPLE `cs`h:

```
limit descriptors 128
```

ksh:

```
ulimit -n 128
```

Finalizing Objects in JDK 1.6

If you are developing an Objectivity for Java application using Sun JDK 1.6, you should be aware of the following JDK defect reported in the Sun Developer Network bug database:

Java objects created in JNI using `AllocObject` are not finalized

This defect affects your application if any of your persistence-capable classes defines a `finalize()` method.

The workaround is to add a default constructor that simply creates an empty object of the class.

EXAMPLE

```
public class PCclass extends ooObj {

    //...

    // If this exists...
    protected void finalize() {
    }

    // ... you must add this...
    public PCclass() { // Construct a dataless object
    }

}
```

Java Command Line Options

Stack Size

The default maximum native stack size allocated by the Java virtual machine for any thread (including the main thread) is platform-specific. In general, these default values are inadequate for Objectivity for Java. An inadequate stack size may cause an Objectivity for Java application to terminate with a segmentation violation for no apparent reason.

You change the default thread stack size for the Java virtual machine with the option:

```
-Xsssize
```

where

size Number of bytes. To specify kilobytes or megabytes, append *size* with k or m, respectively.

EXAMPLE To specify a stack size of 2 megabytes:

```
% java -Xss2m classname
```

Memory Requirements

If an Objectivity for Java application encounters `java.lang.OutOfMemory` errors, you can increase the amount of memory for the Java virtual machine with either or both of the following options:

`-Xmssize` Sets the initial memory size.

`-Xmxsize` Sets the maximum memory size.

where

size Number of bytes. To specify kilobytes or megabytes, append *size* with k or m, respectively.

HP-UX Architectures

To run an Objectivity for Java application, which uses Objectivity/DB libraries, on either `hprisc` or `hpuxia64`, you must set the `-v2` option. This enables the application to load Objectivity/DB libraries compiled with the `-AA` flag.

64-bit Architectures

To run an application on `sparc64`, `hpuxia64`, or `solaris86_64`, you must specify the `-d64` option. This uses the 64-bit data model of the Java virtual machine instead of the default 32-bit data model.

C

Sample Applications for C++ and Java

A number of sample applications are installed with Objectivity/C++ and Objectivity for Java. You can inspect these samples to see how various types of application tasks might be implemented. You can also compare corresponding samples to see how to accomplish similar tasks in the two Objectivity programming interfaces.

The source files for the sample applications are located in subdirectories of the *installDir\samples* directory:

Table C-1: Location of Samples

Samples for This Programming Interface	Are in Subdirectories of This Directory
Objectivity/C++	<i>installDir/arch/samples/cxx</i>
Objectivity for Java	<i>installDir/arch/samples/java</i>

NOTE The *samples* directory also contains sample applications for demonstrating and testing several other Objectivity products. For information about each of these samples, see the installation chapter or the programming-interface documentation for the product.

Overview of Sample Applications

Table C-2 briefly describes the sample applications and the tasks they perform.

Table C-2: Tasks Performed by Sample Applications

This Subdirectory	Contains an Application That
helloWorld	Connects to and populates a Objectivity/DB federated database; used for verifying Objectivity/C++ and Objectivity for Java installation.
mediaManager	Organizes and finds highly interrelated objects; imports data from a text file; uses Microsoft Foundation Classes (MFC).
networkElement	Manages network elements in a typical telecommunications application.
rentalFleet	Shows how to use basic Objectivity/DB features.
segMap	Builds and uses a segmented map structure; distributes data across databases; clusters data in containers.
tree	Builds and uses a generic tree structure; distributes data across databases; clusters data in containers.
treeView	Shows how to use the Objectivity/C++ classes for ordered (B-tree based) collections.
tutorial	Accompanies the Objectivity/C++ tutorial available through Objectivity/Assist.

See the `readme.txt` file in the `samples/cxx` or `samples/java` directory for details about building the sample applications. See the `readme.txt` file within each application subdirectory for information about executing that sample.

Sample Federated Databases

In some cases, the same federated database is accessed by a pair of corresponding Objectivity/C++ and Objectivity for Java applications. The files for such federated databases are in subdirectories of `installDir/arch/samples/data`.

To work with your own copy of the C++ samples, copy both the `samples/cxx` subdirectory and the `samples/data` subdirectory into your chosen directory. You may need to adjust some pathnames in the sample makefiles.

Similarly, to work with your own copy of the Java samples, copy both the `samples/java` subdirectory and the `samples/data` subdirectory into your chosen directory.

Troubleshooting an Application

The following sections provide some guidelines for fixing problems that may arise when you run an Objectivity/DB application.

Federated Database Does Not Open

Solutions:

- Verify that the `OO_FD_BOOT` environment variable is set to the path of the boot file, or that the full pathname for the boot file is correct.
- Check the network node specified for the lock server in the boot file to make sure that `ooLockServerName` is set to the correct value.
- Verify that the federated database number specified by the `ooFDNumber` value in the boot file is unique.
- Verify that the federated database has a license that is valid for the application you are running.

Lock Server Not Running

Solution:

- Run `oolockmon` to check whether a lock server is running. If necessary, run `oolockserver` to start the lock server on your machine.

Object Does Not Open

Solutions:

- Verify that a lock server is running on the node specified by the `ooLockServerName` value in the boot file.
- Check whether a network failure is preventing access to the node where the lock server is running.

- If a dbx session was terminated while debugging an application, check if any locks remain.
- If your application disabled the locking mechanism, make sure that other applications are not accessing the same data.

Lock Server Timed Out

Solutions:

- Consider moving the lock server to a less congested host.
- Consider increasing the network timeout period by setting the `OO_RPC_TIMEOUT` environment variable to the desired number of seconds (greater than the default of 25 seconds).
- If you are using NFS, consider decreasing the NFS data packet size by setting the `OO_NFS_MAX_DATA` environment variable to the desired number of bytes (less than the default of 8192 bytes).

Index

A

- accessing installation files 15
- Active Schema
 - for C++
 - installing 41
 - library 104
 - preparing to use 42
 - release files 43
 - system requirements 41
 - for Java
 - installing 51
 - preparing to use 52
 - release files 52
 - system requirements 51
- adjusting data packet size 27
- Advanced Multithreaded Server (see AMS)
- AMS
 - log directory 28
 - setting up 27
 - using with Objectivity/HA 87
- application
 - compiling 106
 - linking 99
 - programming issues 107
- arch abbreviation 16
- architectures, UNIX 14
- Assist 22
 - configuring your own Eclipse for 23
 - software requirements 14
 - starting 22
 - workspace directory 22
- autonomous partitions 87

C

- CD, mounting 15
- CLASSPATH environment variable 47
- client host 26
- compiling
 - compiler flags 106
 - Objectivity/C++ application 106
- Customer Support 11
- customizing the python executable 62–65

D

- Data Definition Language 35
- data packet size, used with NFS 27
- data replication option (see Objectivity/HA)
- data server
 - host 26
 - software 26
- data source
 - adding for federated database 81
 - personal or system 81
- database images 87
- DDL
 - files 35
 - processor 35
 - configuring 36
- debugging
 - compiling and linking for 108
 - printing
 - handles 108
 - objects 109

demo applications

- Interactive SQL++ 73
- Objectivity for Java 48, 117
- Objectivity/C++ 38, 117
- Objectivity/ODBC 84
- Objectivity/Python 61

Dharma/SQL 67

- documentation 70

distribution CD, mounting 15

download installation files 15

driver-manager, ODBC 77

DRO (see Objectivity/HA)

E

Eclipse Platform

- configuring for Assist 23

Eclipse Rich Client Platform 22

environment variables

- CLASSPATH 47
- LD_LIBRARY_PATH 18
- OO_FD_BOOT 121
- OO_NFS_MAX_DATA 27, 122
- OO_RPC_TIMEOUT 122
- OO_SQL_DIR 69, 79, 80, 81, 82
- PATH 18, 23
- PYTHONPATH 61
- SHLIB_PATH 18
- XBMLANGPATH 31
- XFILESEARCHPATH 31

ENVY/Developer

- requirements 53
- setting up 56

errors

- federated database 121
- file descriptor 107, 112
- linking 103
- lock server 33, 121
- running an application 121
- verifying installation 31

F

fault tolerant option (see Objectivity/HA)

federated database 13

- adding data sources 81
- errors 121
- license in 21

file-descriptor limit

- specifying for C++ process 107
- specifying for Java process 112

FTO (see Objectivity/HA)

H

HA abbreviation 10

high availability (see Objectivity/HA)

hostname for ODBC server 83

hprisc

- architecture 14
- link rules 100, 101, 103

hpuxia64

- architecture 14
- link rules 100, 101, 103

I

installation files, accessing 15

installation, troubleshooting 31

installDir abbreviation 16

installing

- Active Schema for C++ 41
- Active Schema for Java 51
- Objectivity for Java 45
- Objectivity/C++ 35
- Objectivity/DB 13
- Objectivity/DDL 35
- Objectivity/HA 87
- Objectivity/IPLS 95
- Objectivity/ODBC 77
- Objectivity/PQE 91
- Objectivity/Python 59
- Objectivity/Smalltalk for VisualWorks 53
- Objectivity/SQL++ 67

Interactive SQL++

- defined 67
- sample applications 73
- testing 73

interoperating between releases 19**IPLS abbreviation 10****L****LD_LIBRARY_PATH environment variable**

18

liboo.a 100**liboo_adm.a 100****liboo_dbx.a 100****liboolspm.a 105****library**

- Active Schema 104
- administration 100
- debug 100
- Objectivity/C++ persistent collections 105
- Objectivity/DB
 - shared runtime 101
 - static runtime 100
- Objectivity/IPLS 96, 104
- reentrant C 106
- standard 100

license 21**linking lock-server performance-monitoring program 105****linking Objectivity/C++ application 99**

- link rules
 - development environments 101
 - end-user deployment 103
 - static linking 100
- with Active Schema 104
- with Objectivity/IPLS 104
- with persistent collections 105
- with Purify 103

linux86_64

- architecture 14
- link rules 100, 102, 103

linux86gcc3

- architecture 14
- link rules 100, 102, 103

lock server 24

- errors 33, 121
- in-process 95
- log directory 24
- performance-monitoring program, linking 105
- port 24
- setting up 24

log directory, setting up

- for AMS 28
- for in-process lock server 96
- for lock server 25
- for ODBC server 72

M**makefile**

- Interactive SQL++ sample 73
- Objectivity/C++ sample 106

memory, increasing for Java 114**monitoring lock-server performance 105****mounting the distribution CD 15****N****Network File System (NFS) 26**

- data packet size 27, 122
- setting up 26

network timeout period, setting 122**O****Objectivity for Java**

- compiler requirements 45
- increasing memory 114
- installing 45
- preparing to use 47
- release files 48
- sample applications 48, 117
- specifying file-descriptor limit 112
- specifying stack size 114
- system requirements 45
- testing 48

Objectivity for Java with Java Connection Architecture (JCA) 45

Objectivity servers

- AMS 26
- lock server 24
- ODBC server 70
- query server 93

Objectivity/Assist (see Assist)**Objectivity/C++**

- compiler requirements 35
- compiling 106
- debugging application 108
- installing 35
- linking application 99
- persistent collections libraries 105
- persistent collections library 105
- preparing to use 37
- programming issues 107
- release files 38
- sample applications 38, 117
- specifying file descriptor limit 107
- system requirements 35
- testing 38

Objectivity/DB

- accessing installation files 15
- installing 13
- preparing to use 18
- release files 20
- shared runtime library 101
- static runtime library 100
- system requirements 14
- upgrading existing federated databases 19

Objectivity/DB Active Schema

(see Active Schema)

Objectivity/DB High Availability (see**Objectivity/HA)****Objectivity/DB In-Process Lock Server**

(see Objectivity/IPLS)

Objectivity/DDL

- installing 35
- preparing to use 37
- release files 38
- system requirements 35
- testing 38

Objectivity/DRO (see Objectivity/HA)**Objectivity/FTO (see Objectivity/HA)****Objectivity/HA 87**

- installing 87
- preparing to use 89
- release files 89
- system requirements 87

Objectivity/IPLS

- installing 95
- library 96, 104
- loading shared library 97
- log directory 96
- preparing to use 96
- release files 96
- system requirements 95

Objectivity/ODBC

- adding data sources 81
- configuring TCP/IP 83
- installing 77
- ODBC driver-manager 77
- preparing to use 79
- registering with driver-manager 80
- release files 79
- sample client application 84
- software requirements 77
- supported ODBC standard 77
- system requirements 77
- TCP/IP port 84
- testing 84

Objectivity/Parallel Query Engine (see**Objectivity/PQE)****Objectivity/PQE 91**

- installing 91
- loading shared library 94
- preparing to use 92
- release files 93
- system requirements 91

Objectivity/Python 59

- customizing the python executable 62
- installing 59
- preparing to use 61
- release files 61
- sample applications 61
- system requirements 59

Objectivity/Smalltalk for VisualWorks

- installing 53
- preparing to use 54
- release files 55
- system requirements 53
- testing 56

Objectivity/SQL++

- Dharma/SQL version embedded 67
- installing 67
- Interactive SQL++
 - defined 67
 - testing 73
- log directory, setting up 72
- ODBC server
 - defined 67
 - registering port number 84
 - setting up 70
 - specifying hostname 83
 - TCP/IP port 71
 - testing 74
- preparing to use 69
- release files 70
- supported ODBC standard 67
- system requirements 67
- testing
 - Interactive SQL++ 73
 - ODBC server 74

**Objectivity/SQL++ ODBC Driver
(see Objectivity/ODBC)****ODBC**

- driver (see Objectivity/ODBC)
- server (see Objectivity/SQL++)

odbcinst 81, 83**ODMG abbreviation** 10**online books**

- location 20

OO_FD_BOOT environment variable 121**OO_NFS_MAX_DATA environment variable**
27, 122**OO_RPC_TIMEOUT environment variable**
122**OO_SQL_DIR environment variable** 69, 79,
80, 81, 82**ooconfig script** 36**ooddix** 36**oolockserver** 24**oosqld** 71**oosqlnw service** 71, 84**oostartams** 28**ootoolmgr, setting up** 29**ooverify**

- errors 31
- testing installation 17

P**packet size, used with NFS** 27**PATH environment variable** 18, 23**persistent collections**

- libraries for 105
- linking Objectivity/C++ applications 105

port number

- conflict 24, 28
- registering, for ODBC server 84

PQE (Objectivity/PQE)**PQE abbreviation** 10**predefined signal handler** 107, 111**preparing to use**

- Active Schema for C++ 42
- Active Schema for Java 52
- Objectivity for Java 47
- Objectivity/C++ 37
- Objectivity/DB 18
- Objectivity/DDL 37
- Objectivity/HA 89
- Objectivity/IPLS 96
- Objectivity/ODBC 79
- Objectivity/PQE 92
- Objectivity/Python 61
- Objectivity/Smalltalk for VisualWorks 54
- Objectivity/SQL++ 69

printing while debugging

- handles 108
- persistent objects 109

Purify, linking with 103**python executable, customizing** 62**PYTHONPATH environment variable** 61

Q

query server 93

R

reentrant C libraries 106

registering Objectivity/ODBC with a driver-manager 80

releases, interoperating between 19

RPC timeout error 27

(see also network timeout period)

running a multithreaded application

C++ 107

Java 112

S

sample applications

Interactive SQL++ 73

Objectivity for Java 48, 117

Objectivity/C++ 38, 117

Objectivity/ODBC 84

Objectivity/Python 61

setting up

AMS 27

Eclipse Platform for Assist 23

lock server 24

NFS 26

Objectivity/SQL++ ODBC server 70

ootoolmgr 29

query server 93

VisualWorks 55

VisualWorks with ENVY/Developer 56

SHLIB_PATH environment variable 18

signal handler, predefined 107, 111

signals 107, 111

solaris7

architecture 14

link rules 100, 102, 103

solaris86_64

architecture 14

link rules 101, 102, 103

sparc64

architecture 14

link rules 101, 102, 103

stack size

specifying for C++ 107

specifying for Java 114

supported ODBC standard 77

syste 69, 71

T

TCP/IP

configuring for Objectivity/ODBC 83

services file 84

troubleshooting

applications 121

installation 31

typographical conventions 10

U

unixODBC DriverManager 77

UNIX-specific development issues

Objectivity for Java 111

Objectivity/C++ 99

upgrading federated databases

for Objectivity/DB 19

V

VisualWorks

requirements 53

setting up 55

X

X Window System 14

resources for ootoolmgr 29

XBMLANGPATH environment variable 31

XFILESEARCHPATH environment variable
31