



# Installation and Platform Notes for UNIX

Release 5.2

## Installation and Platform Notes for UNIX

Part Number: 52-IUNIX-0

Release 5.2, September 29, 1999

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 1999 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. Objectivity/DB Fault Tolerant Option, Objectivity/FTO, Objectivity/DB Data Replication Option, Objectivity/DRO, Objectivity/DB Hot Failover, Objectivity/DB Open File System, Objectivity/OFS, Objectivity/DB Secure Framework, Objectivity/Secure, Objectivity/C++, Objectivity/C++ Data Definition Language, Objectivity/DDL, Objectivity/C++ Active Schema, Objectivity/C++ Standard Template Library, Objectivity/C++ STL, Objectivity/C++ Spatial Index Framework, Objectivity/Spatial, Objectivity for Java, Objectivity/Smalltalk, Objectivity/SQL++, Objectivity/SQL++ ODBC Driver, Objectivity/ODBC, and Objectivity Event Notification Services are trademarks of Objectivity, Inc. Standards<ToolKit> is a trademark of ObjectSpace, 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.

U. S. Government Restricted Rights: Use, duplication or disclosure of the software or other information by the U. S. Government or any unit or agency thereof is subject to restrictions as set forth in subparagraph (c) (1) (ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 and the Government is acquiring only restricted rights in the software and limited rights in any technical data provided (as such terms are defined in such clause of the DFARS). If the software or other information is supplied to any unit or agency of the U. S. other than the Department of Defense, the Government's rights will be as defined in clause 52.227-19(c)(2) of the FAR or, in the case of NASA, in clause 18-52.227-86 (d) of the NASA Supplement to the FAR.

# Contents

---

<b>About This Book</b>	<b>7</b>
Audience	7
Organization	7
Conventions and Abbreviations	8
Getting Help	9
<b>Chapter 1 Objectivity/DB Installation</b>	<b>11</b>
System Requirements	11
Installing Objectivity/DB	12
Setting Up the Lock Server	15
Setting Up Data Server Software	16
Setting Up Objectivity/DB Graphical Tools	17
Upgrading Existing Federated Databases	19
Troubleshooting Installation	24
<b>Chapter 2 Objectivity/C++ Installation</b>	<b>27</b>
System Requirements	27
Installing Objectivity/C++ or Objectivity/DDL	28
Testing Objectivity/C++ Setup	30
<b>Chapter 3 Objectivity/C++ STL Installation</b>	<b>33</b>
System Requirements	33
Installing Objectivity/C++ STL	33
Testing Objectivity/C++ STL Setup	35

<b>Chapter 4</b>	<b>Objectivity/C++ Active Schema Installation</b>	<b>37</b>
	System Requirements	37
	Installing Objectivity/AS	37
<b>Chapter 5</b>	<b>Objectivity for Java Installation</b>	<b>41</b>
	System Requirements	41
	Installing Objectivity for Java	42
	Upgrading a Release 4.0.10 Federated Database	44
	Testing Objectivity for Java Setup	44
<b>Chapter 6</b>	<b>Objectivity/Smalltalk for VisualWorks Installation</b>	<b>45</b>
	System Requirements	45
	Installing Objectivity/Smalltalk for VisualWorks	46
	Setting Up VisualWorks	48
	Setting Up VisualWorks With ENVY/Developer	49
	Testing Objectivity/Smalltalk for VisualWorks Setup	50
<b>Chapter 7</b>	<b>Objectivity/SQL++ Installation</b>	<b>51</b>
	System Requirements	51
	Installing Objectivity/SQL++	52
	Setting Up the Objectivity/SQL++ ODBC Server	55
	Testing Interactive SQL++	56
	Testing the Programming Interface	57
	Preparing the ODBC Server for Testing	58
<b>Chapter 8</b>	<b>Objectivity/FTO Installation</b>	<b>61</b>
	System Requirements	61
	Installing Objectivity/FTO	61
	Setting Up Objectivity/Smalltalk for VisualWorks	63
<b>Chapter 9</b>	<b>Objectivity/DRO Installation</b>	<b>65</b>
	System Requirements	65
	Installing Objectivity/DRO	66
	Setting Up Objectivity/Smalltalk for VisualWorks	67

<b>Appendix A C++ Application Development</b>	<b>69</b>
Linking User Applications With Objectivity/DB	69
Libraries for Static Linking	69
Linking to Shared Libraries	70
Linking With Purify	72
Linking Under AIX	72
Linking to Additional Objectivity Products and Features	73
Using Makefiles	74
Objectivity/C++ Applications	74
Objectivity/C++ STL Applications	74
Application Programming Issues	74
Signal Handling	74
Stack Size for Multithreaded Applications	75
File Descriptor Limit	75
Debugging an Application	75
Preparing to Debug an Application	76
Printing Handles	76
Printing Objects	76
<b>Appendix B Java Application Development</b>	<b>77</b>
Running an Objectivity for Java Application	77
Changing the Stack Size	77
File Descriptor Limit	78
Memory Requirements	78
<b>Appendix C Troubleshooting an Application</b>	<b>79</b>
<b>Index</b>	<b>81</b>



# About This Book

---

This book, *Installation and Platform Notes for UNIX*, describes how to install Objectivity products on 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. It 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 C++ or Java database applications on UNIX platforms. These 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++ book and the Objectivity/C++ Standard Template Library book. 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 online Objectivity for Java guide.

# Conventions and Abbreviations

## Navigation

Table of contents entries, index entries, cross-references, and underlined text are hypertext links.

## Typographical Conventions

<code>oobackup</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
<b>Browse FD</b>	Graphical user-interface label for a menu item or button
<code>lock server</code>	New term, book title, or emphasized word

## Abbreviations

<i>(administration)</i>	Feature intended for database administration tasks
<i>(FTO)</i>	Objectivity/DB Fault Tolerant Option feature
<i>(DRO)</i>	Objectivity/DB Data Replication Option feature
<i>(ODMG)</i>	Feature conforming to the Object Database Management Group interface

## Command Syntax Symbols

<code>[...]</code>	Optional item. You may either enter or omit the enclosed item.
<code>{...}</code>	Item that can be repeated.
<code>... ...</code>	Alternative items. You should enter only one of the items separated by this symbol.
<code>(...)</code>	Logical group of parameters. 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 (labelled 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. Call Objectivity Customer Support to get access to the site.

### How to Reach Objectivity Customer Support

You can contact Objectivity Customer Support by:

- **Telephone:** Call 1.650.254.7100 or 1.800.SOS.OBJY (1.800.767.6259) Monday through Friday between 6 AM and 6 PM 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 1.650.254.7171.
- **Electronic Mail:** Send electronic mail to [help@objectivity.com](mailto:help@objectivity.com).

### Before You Call

If you need help from Customer Support, please have the following information ready before you contact Objectivity:

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



## Objectivity/DB Installation

---

This chapter describes the requirements and steps for installing Objectivity/DB on a UNIX platform. Objectivity/DB provides the object database services that enable your applications to create and access persistent objects. Objectivity/DB also provides tools for database administration and data inspection.

### System Requirements

You can install Objectivity/DB on the UNIX architectures listed in Table 1-1. Each architecture represents a combination of hardware and UNIX operating system.

---

**NOTE** See the online release notes on the Objectivity Technical Support web site for the currently supported operating-system versions. Contact Objectivity Customer Support to get access to this web site.

---

**Table 1-1:** Supported UNIX Architectures

Hardware	Operating-System Version	Architecture Name
DEC Alpha	<i>See web site</i>	alphaosf1
HP 9000 Series 700/800	HP/UX 10.20 HP/UX 11.0	hp9000s700 hprisc
IBM RISC System/6000	<i>See web site</i>	ibmrs6000
Intel x86	<i>See web site</i>	linux86
Silicon Graphics IRIS	<i>See web site</i>	iris
SPARCstation	Solaris 2.6 Solaris 7	solaris4 solaris7

## Software Requirements

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

- C++ runtime library (required for the Objectivity/DB kernel and tools)
- X Window System (required for running Objectivity/DB graphical tools)

## Viewing Online Books

To view Objectivity online books in Portable Document Format (PDF), you must install the freely available Acrobat Reader software from Adobe Systems, Inc. You can obtain an Acrobat Reader from Adobe's online services. Use your World Wide Web browser to access the web site [www.adobe.com](http://www.adobe.com).

## Hardware Requirements

Your workstation must have a CD-ROM drive that is either local or accessible on the network.

## Installing Objectivity/DB

To install the release files for Objectivity/DB, either alone or in combination with one or more other products:

1. Log in to your workstation as `root` (required only for mounting the CD-ROM).
2. Verify that your workstation meets the system requirements for installing Objectivity/DB (see "System Requirements" on page 11). If you choose to install other Objectivity products at this time, verify that the requirements for those products are met (see the installation chapter for each product).
3. Locate or create an installation directory for Objectivity/DB. To create the installation directory, enter:

```
mkdir installDir
```

where

*installDir*      Installation directory pathname—for example, `/usr/object`

The installation script will place the release files in a subdirectory of *installDir*—specifically, *installDir/arch*, where *arch* is the architecture name for your platform (see Table 1-1).

You can preserve an existing Objectivity/DB installation by creating a new installation directory (for example, *installDir52*), or by renaming the architecture-specific subdirectory in the existing *installDir* (for example, move *installDir/arch* to *installDir/arch.old*).

**Important:** If you need to upgrade the database format of existing federated databases, you must preserve the installation directory from the earlier release until the upgrade is complete. Upgrading is handled in step 14.

4. Give all Objectivity/DB users both read and execute permission to *installDir*.
5. Insert the Objectivity distribution CD-ROM into the CD-ROM drive and make sure the CD-ROM is mounted. See your operating system documentation for the proper mount command. On the following architectures, use the indicated option to suppress version numbers and case changes in filenames:
  - On the `alphaosf1` architecture, use the `-noversion` option.
  - On the `hp9000s700` and `hprisc` architectures, use the `-o cdcase` option.
6. Run the installation script provided on the distribution CD-ROM.
  - On all architectures except `linux86`, enter:
 

```
cdMountDir/install.csh
```
  - On `linux86`, enter:
 

```
csh cdMountDir/install.csh
```

In these commands:

*cdMountDir*      CD-ROM mount directory—for example, `/cdrom`

7. At the product prompt, specify the product(s) to be installed:
  - To install only Objectivity/DB, enter the item number given for it in the prompt.
  - To install Objectivity/DB with other Objectivity products, enter the corresponding item numbers, separated by commas.

---

**NOTE** You must have a license for each product you want to install.

---

8. At the directory prompt, specify the *installDir* you created in step 3. If you get an error message reporting insufficient disk space or incorrect permissions, you can specify another directory.  
The installation script:
  - Creates the subdirectory *installDir/arch* and copies the product release files into it.
  - Runs the installation verification test `ooverify`, which checks that the release files are loaded properly. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 24.
9. Advise each application developer to add *installDir/arch/bin* to the `PATH` environment variable.

10. Advise each application developer to add `installDir/arch/lib` as the first component of the environment variable indicated below. This is necessary for running Objectivity/DB tools built with shared libraries.
  - On `alphaosf1`, `linux86`, `solaris4`, and `solaris7`, set `LD_LIBRARY_PATH`.
  - On `hp9000s700` and `hprisc`, set `SHLIB_PATH`.
  - On `ibmrs6000`, set `LIBPATH`.
  - On `iris`, set `LD_LIBRARYN32_PATH`.
 

**Note:** `LD_LIBRARY_PATH` can be set instead of `LD_LIBRARYN32_PATH`, but only if `LD_LIBRARYN32_PATH` is not set at all. `LD_LIBRARY_PATH` is ignored when `LD_LIBRARYN32_PATH` is set.
11. Set up the lock server to manage concurrent database access (see “Setting Up the Lock Server” on page 15).
12. Set up data server software (NFS or AMS) to provide remote data access (see “Setting Up Data Server Software” on page 16).
13. Set up Objectivity/DB administration and database tools (see “Setting Up Objectivity/DB Graphical Tools” on page 17).
14. If you plan to access any existing federated databases using tools or applications built with the current Objectivity/DB release, read “Upgrading Existing Federated Databases” on page 19. You may need to upgrade the database format, indexes, or schemas of such federated databases before they can be accessed.
15. If you specified multiple products in step 7, read the installation chapters for those products, and follow any additional steps for setting them up.
16. Read:
  - The printed *Objectivity Release Notes* for new and changed features.
  - The online release notes 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/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 the same 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 (see the “Tools” appendix in the Objectivity/DB administration book)
	Lock server executables (see “Setting Up the Lock Server” below)
	AMS executables (see “Setting Up the Advanced Multithreaded Server” on page 17)
doc	PDF file for the <i>Objectivity/DB Administration</i> online book (see “Viewing Online Books” on page 12). When other Objectivity products are installed, their online books are placed in this subdirectory.
etc	Graphics support for Objectivity/DB tools
	Upgrade file for adding persistent-collection types to pre-Release 5.2 federated database schemas
lib	Shared libraries for Objectivity/DB tools and the Objectivity/DB kernel

## 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 7, “Using a Lock Server,” of the Objectivity/DB administration book.

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 on a lock server host when you start a lock server there for the first time:

1. Log in as `root`.
2. Create the lock server system directory. Enter:
 

```
mkdir /usr/spool/objy
```
3. Set the 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 /usr/spool/objy
```
4. Start the lock server to verify that it is installed properly. Enter:
 

```
installDir/arch/bin/oockserver
```

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

If the lock server still does not start, another process is probably running on the TCP port that the lock server expected to use. If you cannot reassign the other process to a different port, you may have to change the default port for the lock server. To do this, see Chapter 7, “Using a Lock Server,” in the Objectivity/DB administration book.

6. (Optional) Configure your workstation to start the lock server whenever the machine reboots. To do this, add the `oolockserver` command to the workstation’s startup script (usually `/etc/rc.local`).

See Chapter 10, “Automatic and Manual Recovery,” in the Objectivity/DB administration book for information about entering this command with automatic recovery enabled.

## 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 federated database, database, 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 a replicated database (see Chapter 9, “Objectivity/DRO Installation”). For more information about AMS, see the Objectivity/DB administration book.

## Setting Up NFS

Perform the following steps on each UNIX host that is to provide file access to remote database applications through NFS:

1. Find out whether the `nfs` and `mountd` daemons are running. Enter:
 

```
rpcinfo -p hostname
```
2. On the indicated architectures, use the `ps` command to verify that the `mountd` daemon is running with the correct option:
  - The `-n` option on `alphaosf1`, `ibmrs6000`, and `iris`
  - The `-p` option on `hp9000s700` and `hprisc`

3. If necessary, start the `nfs` and `mountd` daemons.
4. Verify that NFS exports any user directories that will contain Objectivity/DB database files. Place these directories in the `/etc/exports` directory.

## Data Packet Size

Your TCP/IP protocol stack may require a smaller data packet size than the default (8192 bytes) used by Objectivity/DB with NFS. In a congested network, an RPC timeout error message may also indicate that the data packet size is too large. You can adjust the data packet size by setting the environment variable `OO_NFS_MAX_DATA`.

## Setting Up the Advanced Multithreaded Server

Perform the following steps on each UNIX host that is to provide file access to remote database applications through AMS. 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.

1. Log in as `root`.
2. Verify that the workstation has a directory called `/usr/spool/objy` (also required for the lock server). If not, create this directory and set its permissions to enable all users to read and write files there.
3. Start AMS to verify that it is installed properly. Enter:

```
installDir/arch/bin/oostartams
```

AMS will not start if another process is running on the TCP port that AMS expected to use. If you cannot reassign the other process to a different port, you may have to change the default port for AMS. To do this, see Chapter 8, “Advanced Multithreaded Server,” in the Objectivity/DB administration book.

4. (Optional) Configure your workstation to start AMS whenever the machine reboots. To do this, add the `oostartams` command to the workstation’s startup script (usually `/etc/rc.local`).

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 Objectivity/DB Graphical Tools

Objectivity/DB provides graphical tools for browsing objects and types in a database. These tools run as X Window System (abbreviated as X) applications. Before you can run these tools, 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 the graphical tools. *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 the graphical tools 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
```

## Upgrading Existing Federated Databases

If you created federated databases with earlier releases of Objectivity/DB, you may need to upgrade them to make them compatible with the current release. Depending on the release that was used to create an existing federated database, you may need to upgrade its database format, indexes, or schema before it can be accessed by tools or applications built with the current release. The following table directs you to the appropriate upgrade procedure.

You Should Upgrade	Created With	See Page
Database format	Release 3.x or Release 4.0.2	20
Indexes	Release 5.0	23
Schema	Any release prior to Release 5.2	23

## Lock Server Compatibility

Tools and applications built with Release 5.2 require a different version of the lock server than is used by prior Objectivity/DB releases. When you upgrade a federated database, you must ensure that its lock server host is running the current lock server. Running an Objectivity/DB application with an incompatible lock server produces an error message.

Multiple versions of the lock server can run on the same workstation so that you can continue to run nonupgraded applications to access nonupgraded federated databases. When a federated database specifies a lock server host that is running multiple lock server versions, you must guarantee that *all* applications accessing the federated database have been built with the *same* release of Objectivity/DB.

---

**WARNING** Data corruption will occur if two applications (built with different releases) contact different lock server versions while accessing data in the same federated database.

---

## Upgrading Database Format

If you created federated databases with Objectivity/DB Release 3.x or Release 4.0.2, and you want to access them using tools or applications built with the current Objectivity/DB release, you must upgrade the federated databases to the current database format.

If you do not know the Objectivity/DB release that was used to create a federated database, you can determine which upgrade procedure to use by running the current release's `ooFile` tool, specifying the federated database file. This will report whether the database format is compatible with Release 3.x or Release 4.x.

---

**NOTE** You can skip this upgrade for federated databases that were created with Release 4.0.10 or later, because they already use the current database format.

---

## Upgrading All Release 3.x Federated Databases

Use this procedure to upgrade *all* your Release 3.x federated databases to the current database format. This procedure kills the current Release 3.x lock server. (To upgrade *some but not all* Release 3.x federated databases, see page 21.)

---

**WARNING** An upgrade cannot be reversed; furthermore, failure of the upgrade tool can irretrievably destroy your federated database. Therefore, you *must* back up your data (step 4). This is the *only* way to protect your data against an upgrade failure.

---

The following steps assume that you have installed the current Objectivity/DB release, and that you have preserved the Objectivity/DB Release 3.x installation.

1. Run the Objectivity/DB Release 3.x `ooLockmon` tool in detail mode to verify that there are no transactions in progress. For example, enter:

```
3.x_InstallDir/arch/bin/ooLockmon -d bootFilePath
```

where

*bootFilePath* Path to the boot file of the Release 3.x federated database. You can omit this parameter if you set the `OO_FD_BOOT` environment variable to the correct path.

2. If there are active transactions, notify the transaction owners either to commit or to abort their transactions.
3. When you are certain that there are no active transactions, stop the Release 3.x lock server using the Release 3.x `ookilllls` tool. For example, to kill the local lock server process, enter:
 

```
3.x_InstallDir/arch/bin/ookilllls
```
4. **Mandatory:** Back up all the federated databases by using the Release 3.x `oobackup` tool, by using `tar`, or by copying the federated database files.
5. If you are upgrading Release 3.0 or 3.5 federated databases, drop all indexes from them (use `ooKeyDesc::dropIndex`). This step is not necessary if you are upgrading from Release 3.8.
6. For each federated database, run the current release's `ooupgrade` tool:
 

```
ooupgrade -standalone bootFilePath
```

 where
 

<code>-standalone</code>	Runs the tool in nonconcurrent mode.
<code>bootFilePath</code>	Same as in step 1.
7. If you are upgrading Release 3.0 or 3.5 federated databases, re-create the indexes that you dropped in step 5 (use `ooKeyDesc::createIndex`).

### **Upgrading Selected Release 3.x Federated Databases**

If you want to upgrade *some but not all* of your Release 3.x federated databases, you can adapt the steps for upgrading all Release 3.x federated databases as follows:

- Omit step 3 (that is, you can leave the Release 3.x lock server running).
- In step 4, you back up only the federated databases to be upgraded.
- In step 6, you run the current release's `ooupgrade` tool *without* the `-standalone` option, and only for the federated databases to be upgraded.

### **Upgrading All Release 4.0.2 Federated Databases**

Use this procedure to upgrade *all* your Release 4.0.2 federated databases to the current database format. This procedure kills the current Release 4.0.2 lock server. (To upgrade *some but not all* Release 4.0.2 federated databases; see page 22.)

---

#### **WARNING**

An upgrade cannot be reversed. Furthermore, failure of the upgrade tool can irretrievably destroy your federated database. Therefore, you *must* back up your data (step 3). This is the *only* way to protect your data against an upgrade failure.

---

The following steps assume that you have installed the current Objectivity/DB release, and that you have preserved the Objectivity/DB Release 4.0.2 installation.

1. Run the Objectivity/DB Release 4.0.2 `oolockmon` tool in detail mode to verify that there are no active transactions. For example, enter:

```
4.0.2_InstallDir/arch/bin/oolockmon -d bootFilePath
```

where

*bootFilePath* Path to the boot file of the Release 4.0.2 federated database. You can omit this parameter if you set the `OO_FD_BOOT` environment variable to the correct path.

If there are active transactions, notify the transaction owners either to commit or to abort their transactions.

2. When you are certain that there are no active transactions, stop the Release 4.0.2 lock server using the Release 4.0.2 `ookillls` tool. For example, to kill the local lock server process, enter:

```
4.0.2_InstallDir/arch/bin/ookillls
```

3. **Mandatory:** Back up all the federated databases by using the Release 4.0.2 `oobackup` tool, by using `tar`, or by copying the federated database files.

4. For each federated database, run the current release's `ooupgrade402` tool:

```
ooupgrade402 -standalone bootFilePath
```

where

`-standalone` Runs the tool in nonconcurrent mode.

*bootFilePath* Same as in step 1.

### **Upgrading Selected Release 4.0.2 Federated Databases**

If you want to upgrade *some but not all* of your Release 4.0.2 federated databases, you adapt the steps for upgrading all Release 4.0.2 federated databases as follows:

- Omit step 2 (that is, you can leave the Release 4.0.2 lock server running).
- In step 3, you back up only the federated databases to be upgraded.
- In step 4, you run the current release's `ooupgrade402` tool *without* the `-standalone` option, and only for the federated databases to be upgraded.

## Upgrading Index Format

If you used Release 5.0 to create indexes in a federated database, you must convert these indexes to the current release's format.

---

**NOTE** You can skip this section for indexes created before or after Release 5.0. (Indexes created in Release 3.0 and 3.5 are converted when you upgrade database formats.)

---

- To upgrade Release 5.0 indexes, you can either:
  - Contact Objectivity Customer Support to obtain an index conversion program.
  - Create and run a program that drops and re-creates each Release 5.0 index. If you write this program in Objectivity/C++, you must explicitly delete and re-create every `ooKeyDesc` and `ooKeyField` object.

## Upgrading Schemas

In general, the schema format used by the current Objectivity/DB release is compatible with the schema format of prior Objectivity/DB releases, so you do not need to reprocess your schema files. However, if you want to store persistent collections in a federated database that was created prior to Release 5.2, you must upgrade its schema. You must perform this upgrade before you create or access persistent collections from an application written in Objectivity/C++, Objectivity for Java, or Objectivity/Smalltalk for VisualWorks.

- For each existing federated database that is to store persistent collections, enter the following command:

```
ooschemaupgrade installDir/arch/etc/collectionSchema.dmp
                bootFilePath
```

where

*bootFilePath* Path to the boot file of the federated database. You can omit this parameter if you set the `OO_FD_BOOT` environment variable to the correct path.

# 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 loaded 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-ROM. Rerun `ooverify` (enter `installDir/arch/bin/ooverify`). If the problem persists, your distribution CD-ROM may have been damaged. Contact Objectivity Customer Support for assistance.

## **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...
The manifest file does not contain an entry for itself
```

**Causes:** After the product release files are loaded onto your disk, `ooverify` finds that the file and directory names in your installation directory match those on the distribution CD-ROM; 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-ROM. Rerun `ooverify` (enter `installDir/arch/bin/ooverify`). If the problem persists, your distribution CD-ROM may be incorrect. Contact Objectivity Customer Support for assistance.

## ooverify: File or Directory Permission and File System Errors

### Error messages produced by ooverify:

```
Wrong access mode
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 loaded 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-ROM. Rerun ooverify (enter `installDir/arch/bin/ooverify`). If the problem persists, you may be following the installation procedure incorrectly. Contact Objectivity Customer Support for assistance.

## Lock Server: File or Directory Permission Errors

### Error message produced when starting the lock server:

```
ools:Error in opening /usr/spool/objy/ooRsvTbx
O.S. errno = 13 -- permission denied
Failed to connect to server...
```

**Cause:** Objectivity/DB cannot access the current version of the reserve locks file in the `/usr/spool/objy` directory.

**Solutions:** Either:

- Use `chmod` to set less restrictive access privileges for the reserve locks file (\* stands for the current version number for this file):
 

```
/usr/spool/objy/ooRsvTb*
```
- Start the lock server as the root user.

## Lock Server: File System Errors

**Error message produced when starting the lock server:**

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

**Cause:** Objectivity/DB cannot find the /usr/spool/objy directory.

**Solution:** Perform steps 2 and 3 on page 15 to create the /usr/spool/objy directory.

**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 an Objectivity/DB database.
- Objectivity/DDL is a preprocessor for converting Data Definition Language (DDL) files into a schema of persistent C++ data types in an Objectivity/DB 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 11.

## Software Requirements

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

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

- Objectivity/DB (see Chapter 1)
- The vendor-supplied C++ compiler for your architecture

**Note:** See the online release notes 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.

## Hardware Requirements

Your workstation must have a CD-ROM drive that is either local or accessible on the network.

## Installing Objectivity/C++ or Objectivity/DDL

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

1. Log in to your workstation as `root` (required only for mounting the CD-ROM).
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 27).
3. Insert the Objectivity distribution CD-ROM into the CD-ROM drive and make sure the CD-ROM is mounted. On the following architectures, use the indicated option to suppress version numbers and case changes in filenames:
  - On the `alphaosf1` architecture, use the `-noversion` option.
  - On the `hp9000s700` and `hprisc` architectures, use the `-o cdcase` option.
4. Run the installation script provided on the distribution CD-ROM.
  - On all architectures except `linux86`, enter:
 

```
cdMountDir/install.csh
```
  - On `linux86`, enter:
 

```
csh cdMountDir/install.csh
```

In these commands:

`cdMountDir`      CD-ROM mount directory—for example, `/cdrom`

5. At the product prompt, specify the product(s) to be installed:
  - To install just Objectivity/C++, enter the item number given for it in the prompt.
  - 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 have a license for each product you want to 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 (see Table 1-1 on page 11).
  - Runs the installation verification test `ooverify`, which checks that the release files are loaded properly. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 24.
  - Runs the configuration script `ooconfig` (Objectivity/DDL installation only). The `ooconfig` script prompts you for compiler and environment information, configures the DDL processor, and creates the DDL processor executable (`ooddlx`).
7. (Optional) Test the installation of Objectivity/C++ and Objectivity/DDL by running the provided C++ demo applications (see “Testing Objectivity/C++ Setup” on page 30).
  8. If you intend to store Objectivity/C++ persistent collections in federated databases created prior to Release 5.2, make sure you have upgraded the schemas of those federated databases (see “Upgrading Schemas” on page 23).
  9. Read:
    - Appendix A, “C++ Application Development,” for platform-specific information about using Objectivity/C++.
    - The printed *Objectivity Release Notes* for new and changed features.
    - The online release notes 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 ( <code>ooconfig</code> ) for configuring the DDL processor; executable ( <code>ooddlx</code> ) created by <code>ooconfig</code> .
demo	C++ database applications for demonstration and testing (see “Testing Objectivity/C++ Setup” on page 30).
doc	PDF file for the <i>Objectivity/C++ Data Definition Language</i> online book (see “Viewing Online Books” on page 12).

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

Subdirectory	Contains
include	Include files for the C++ programming interface.
lib	Static libraries for linking C++ applications (see “Linking User Applications With Objectivity/DB” on page 69). Shared library for persistent collections (see “Linking to Additional Objectivity Products and Features” on page 73).

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 C++ demo applications provided with the installation. These applications generate a federated database and interact with it through the C++ interface. You can also inspect the demo applications to see how to use various Objectivity/C++ features.

To build and run the C++ demo applications, follow these steps:

1. Copy the demo directory 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 -r installDir/arch/demo/CC homeDir/CC_demo
cd homeDir/CC_demo
```
2. Edit the makefile to set variables as appropriate to your installation. The lines containing these variables are near the beginning of the file:
  - 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 (see “Setting Up the Lock Server” on page 15)—for example:
 

```
LS_HOST = myLockServerHost
```
  - Set `FDID` to a unique federated database identifier (a number from 1 to 32,000)—for example:
 

```
FDID = 4567
```
3. Start the lock server. Enter:
 

```
installDir/arch/bin/oockserver
```

4. Execute `make` to create the demo federated database and to build three demo applications (`import`, `modify`, and `export`). Enter:

```
make
```

5. Run the demo applications through the `demo.sh` script. Enter:

```
./demo.sh
```

If Objectivity/C++ is installed correctly, you will see these messages:

```
Importing data into the database...
```

```
Modifying the database...
```

```
Exporting data from the database...
```

```
Comparing the result...
```

```
Test PASSED -- The expected results are achieved.
```

If other messages are displayed, inspect the makefile, the compiler, and the settings of the environment variables. Correct any errors and run the demo application again. If there are no installation, `make`, or compiler errors, and the application still fails, contact Objectivity Customer Support for assistance.



## Objectivity/C++ STL Installation

---

This chapter describes the requirements and steps for installing Objectivity/C++ Standard Template Library (Objectivity/C++ STL) on a UNIX platform. Objectivity/C++ STL is an extension of the ObjectSpace Standards<ToolKit> STL implementation. Objectivity/C++ STL adds persistence to ObjectSpace STL classes so your application can store STL class objects in an Objectivity/DB database.

### System Requirements

You can install Objectivity/C++ STL on any of the UNIX architectures listed in Table 1-1 on page 11.

### Software Requirements

Objectivity/C++ STL requires that the following software be installed on your system:

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

### Hardware Requirements

Your workstation must have a CD-ROM drive that is either local or accessible on the network.

### Installing Objectivity/C++ STL

To install Objectivity/C++ STL:

1. Log in to your workstation as `root` (required only for mounting the CD-ROM).

2. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 33).
3. Insert the Objectivity distribution CD-ROM into the CD-ROM drive and make sure the CD-ROM is mounted. See your operating system documentation for the proper mount command. On the following architectures, use the indicated option to suppress version numbers and case changes in filenames:
  - On the `alphaosf1` architecture, use the `-noversion` option.
  - On the `hp9000s700` and `hprisc` architectures, use the `-o cdcase` option.
4. Run the installation script provided on the distribution CD-ROM.
  - On all architectures except `linux86`, enter:  
`cdMountDir/install.csh`
  - On `linux86`, enter:  
`csh cdMountDir/install.csh`

In these commands:

`cdMountDir`      CD-ROM mount directory—for example, `/cdrom`

5. At the product prompt, specify Objectivity/C++ STL by typing its item number from the displayed list.

---

**NOTE** You must have a license for Objectivity/C++ STL to install it.

---

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:
  - Creates the `ToolKit` subdirectory in `installDir/arch`, where `arch` is the architecture name for your platform, and places the release files in subdirectories of `ToolKit` (see Table 3-1).
  - Runs the installation verification test `ooverify`, which checks that the release files are loaded properly. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 24.
7. (Optional) Test the installation of Objectivity/C++ STL by running the provided C++ demo applications (see “Testing Objectivity/C++ STL Setup” on page 35).
8. Read:
  - Appendix A, “C++ Application Development,” for platform-specific information about compiling and linking this product.
  - The printed *Objectivity Release Notes* for new and changed features.

- The online release notes 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/C++ STL

When you install Objectivity/C++ STL, its files are organized in subdirectories of the `installDir/arch/ToolKit` directory, as shown in Table 3-1.

**Table 3-1:** Objectivity/C++ STL Release Files in `installDir/arch/ToolKit`

Subdirectory of ToolKit	Contains
<code>config</code>	Configuration file ( <code>local.cfg</code> ) to be included in makefiles (defines compile and link flags)
<code>doc</code>	ObjectSpace STL online documentation
<code>lib</code>	Static libraries for linking applications with ObjectSpace STL and Objectivity/C++ STL (see “Linking to Additional Objectivity Products and Features” on page 73)
<code>ospace/stl</code>	Include files and source files for ObjectSpace STL and Objectivity/C++ STL
<code>ospace/stl/d_examples</code>	Objectivity/C++ STL applications for demonstration and testing

The installation script also places the PDF file for the *Objectivity/C++ Standard Template Library* online book in the `installDir/arch/doc` directory. To view this book, see “Viewing Online Books” on page 12.

## Testing Objectivity/C++ STL Setup

You can test whether Objectivity/C++ STL is set up correctly by building and running the demo applications provided with the installation. You can also inspect the demo applications to see how to use various Objectivity/C++ STL features.

To build and run the STL demo applications, follow these steps:

1. Edit the configuration file `ToolKit/config/local.cfg` to set the `TOOLKIT` variable to the location of the `ToolKit` subdirectory:
 

```
TOOLKIT = installDir/arch/ToolKit
```
2. Change your working directory to the STL demo directory. Enter:
 

```
cd installDir/arch/ToolKit/ospace/stl/d_examples
```

3. Edit the makefile in the demo directory to set variables as appropriate to your installation. The lines containing these variables are near the beginning of the file:
  - Set `OBJECTIVITY_ROOT` to the location of the Objectivity/DB installation directory *installDir*—for example:  
`OBJECTIVITY_ROOT = /usr/object`
  - Set `LOCK_HOST` to the name of the lock server host (see “Setting Up the Lock Server” on page 15)—for example:  
`LOCK_HOST = myLockServerHost`
  - Set `FD_NUMBER` to a unique federated database identifier (a number from 1 to 32,000)—for example:  
`FD_NUMBER = 4567`
  - Set `OSPACE_DIR` to the location of the STL libraries. Enter:  
`OSPACE_DIR = $(TOOLKIT)/lib`
4. Start the lock server. Enter:  
`installDir/arch/bin/oologserver`
5. Enter the following command to create the demo federated database and then build and run the demo applications:  
`make tests`

If error messages are displayed, verify the makefile, the compiler, and the settings of the environment variables. Correct any errors and run the demo application again. If there are no installation, `make`, or compiler errors, and the application still fails, contact Objectivity Customer Support for assistance.

## Objectivity/C++ Active Schema Installation

---

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

### System Requirements

You can install Objectivity/AS on any of the UNIX architectures listed in Table 1-1 on page 11.

### Software Requirements

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

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

### Hardware Requirements

Your workstation must have a CD-ROM drive that is either local or accessible on the network.

### Installing Objectivity/AS

To install Objectivity/AS:

1. Log in to your workstation as `root` (required only for mounting the CD-ROM).
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 37).

3. Insert the Objectivity distribution CD-ROM into the CD-ROM drive and make sure the CD-ROM is mounted. See your operating system documentation for the proper mount command. On the following architectures, use the indicated option to suppress version numbers and case changes in filenames:
  - On the `alphaosf1` architecture, use the `-noversion` option.
  - On the `hp9000s700` and `hprisc` architectures, use the `-o cdcase` option.
4. Run the installation script provided on the distribution CD-ROM.
  - On all architectures except `linux86`, enter:
 

```
cdMountDir/install.csh
```
  - On `linux86`, enter:
 

```
csh cdMountDir/install.csh
```

In these commands:

`cdMountDir`      CD-ROM mount directory—for example, `/cdrom`
5. At the product prompt, specify Objectivity/AS by typing its item number from the displayed list.

---

**NOTE** You must have a license for Objectivity/C++ Active Schema to install it.

---

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 (see Table 1-1 on page 11).
  - Runs the installation verification test `ooverify`, which checks that the release files are loaded properly. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 24.
7. Read:
  - Appendix A, “C++ Application Development,” for platform-specific information about compiling and linking this product.
  - The printed *Objectivity Release Notes* for new and changed features.
  - The online release notes 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/AS

When you install Objectivity/AS, its files are organized in subdirectories of the *installDir/arch* directory, as shown in Table 4-1.

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

Subdirectory	Contains
doc	PDF file for the <i>Objectivity/C++ Active Schema</i> online book (see “Viewing Online Books” on page 12)
include	Include file <code>ooas.h</code> for the Objectivity/AS programming interface
lib	Shared link library for Objectivity/AS (see “Linking to Additional Objectivity Products and Features” on page 73)



## 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 database.

### System Requirements

You can install Objectivity for Java on the UNIX architectures listed in Table 5-1. Each architecture represents a combination of hardware and UNIX operating system.

---

**NOTE** See the online release notes on the Objectivity Technical Support web site for the currently supported operating-system versions. Contact Objectivity Customer Support to get access to this web site.

---

**Table 5-1:** Supported UNIX Architectures

Hardware	Operating-System Version	Architecture Name
DEC Alpha	<i>See web site</i>	alphaosf1
HP 9000 Series 700/800	HP/UX 11.0	hprisc
IBM RISC System/6000	<i>See web site</i>	ibmrs6000
Intel x86	<i>See web site</i>	linux86
Silicon Graphics IRIS	<i>See web site</i>	iris
SPARCstation	Solaris 2.6 Solaris 7	solaris4 solaris7

## Software Requirements

Objectivity for Java requires that the following software be installed:

- Objectivity/DB (see Chapter 1)
- The vendor-supplied Java 2 Software Development Kit (SDK) for your architecture.

**Note:** See the online release notes on the Objectivity Technical Support web site for the currently supported SDK versions. Contact Objectivity Customer Support to get access to this web site.

## Hardware Requirements

Your workstation must have a CD-ROM drive that is either local or accessible on the network.

## Installing Objectivity for Java

To install Objectivity for Java:

1. Log in to your workstation as `root` (required only for mounting the CD-ROM).
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 42).
3. Insert the Objectivity distribution CD-ROM into the CD-ROM drive and make sure the CD-ROM is mounted. See your operating system documentation for the proper mount command. On the following architectures, use the indicated option to suppress version numbers and case changes in filenames:
  - On the `alphaosf1` architecture, use the `-noversion` option.
  - On the `hprisc` architecture, use the `-o cdcase` option.
4. Run the installation script provided on the distribution CD-ROM.
  - On all architectures except `linux86`, enter:
 

```
cdMountDir/install.csh
```
  - On `linux86`, enter:
 

```
csh cdMountDir/install.csh
```

In these commands:

`cdMountDir`      CD-ROM mount directory—for example, `/cdrom`

5. At the product prompt, specify Objectivity for Java by typing its item number from the displayed list.

---

**NOTE** You must have a license for Objectivity for Java to install it.

---

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, and in a subdirectory of your SDK installation directory (see “Release Files for Objectivity for Java” on page 44).
- Runs the installation verification test `ooverify`, which checks that the release files are loaded properly. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 24.

7. Set the `CLASSPATH` environment variable as follows:

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

where

*existingValues* Existing class path components

**Note:** Some application development environments require you to specify `CLASSPATH` from within the tool.

8. Set the `THREADS_FLAG` environment variable as follows:
 

```
setenv THREADS_FLAG native
```

 Objectivity for Java will *not* run with green threads.
9. Upgrade any federated databases that were created with Objectivity/DB Release 4.0.10 if you want to access them using an Objectivity for Java application (see “Upgrading a Release 4.0.10 Federated Database” on page 44).
10. If you intend to store Objectivity for Java persistent collections in federated databases created prior to Release 5.2, make sure you have upgraded the schemas of those federated databases (see “Upgrading Schemas” on page 23).
11. Read:
  - Appendix B, “Java Application Development,” for platform-specific information about using Objectivity for Java.
  - The printed *Objectivity Release Notes* for new and changed features.
  - The online release notes 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 5-2:

**Table 5-2:** Objectivity for Java Release Files in `installDir/arch`

Directory	Files	Contains
doc	java/index.html	Document index <sup>a</sup>
	java/api/*.html	Online reference <sup>a</sup>
	java/guide/*.html	Online guide <sup>a</sup>
	java/guide.pdf	Online guide (PDF) <sup>b</sup>
	java/samples/GettingStarted/*.java	Sample programs
java	src/src.zip	Library source
	lib/oojava.jar	Library executable
lib	liboojava.so	Library executable

a. Viewable through an HTML browser

b. Viewable through Acrobat Reader (see “Viewing Online Books” on page 12)

## Upgrading a Release 4.0.10 Federated Database

If you want to use Objectivity for Java with a federated database that was created with Release 4.0.10, you must upgrade the schema by adding the built-in types specific to Objectivity for Java. To do this, you create an upgrade application that calls the `upgradeSchema4010to50` method of the `objy.db.util.Utility` class. For an example, see the Objectivity for Java reference for this class.

## Testing Objectivity for Java Setup

You can test whether Objectivity for Java is set up correctly by building and running the example application discussed in the “Getting Started” chapter of the Objectivity for Java guide. You can build this application either using a Java IDE or from a command prompt. Before you can run the application, you must create a federated database, as described in the “Getting Started” chapter. You can inspect the example application to see how to use various Objectivity for Java features. The directory containing the sample application is listed in Table 5-2.

## 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 database.

### System Requirements

You can install Objectivity/Smalltalk for VisualWorks on the UNIX architectures shown in Table 6-1. Each architecture represents a combination of hardware and UNIX operating system.

---

**NOTE** See the online release notes on the Objectivity Technical Support web site for the currently supported operating-system versions. Contact Objectivity Customer Support to get access to this web site.

---

**Table 6-1:** Supported UNIX Architectures

Hardware	Operating-System Version	Architecture Name
HP 9000 Series 700/800	HP/UX 11.0	hprisc
Intel x86	<i>See web site</i>	linux86
SPARCstation	2.6 Solaris 7	solaris4 solaris7

## Software Requirements

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

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

**Note:** See the online release notes 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.

## Hardware Requirements

Your workstation must have a CD-ROM drive that is either local or accessible on the network.

## Installing Objectivity/Smalltalk for VisualWorks

To install Objectivity/Smalltalk for VisualWorks:

1. Log in to your workstation as `root` (required only for mounting the CD-ROM).
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 46).
3. Insert the Objectivity distribution CD-ROM into the CD-ROM drive and make sure the CD-ROM is mounted. See your operating system documentation for the proper mount command. On the following architecture(s), use the indicated option to suppress version numbers and case changes in filenames:
  - On the `hprisc` architecture, use the `-o cdcase` option.
4. Run the installation script provided on the distribution CD-ROM.
  - On all architectures except `linux86`, enter:
 

```
cdMountDir/install.csh
```
  - On `linux86`, enter:
 

```
csh cdMountDir/install.csh
```

In these commands:

`cdMountDir`      CD-ROM mount directory—for example, `/cdrom`

5. At the product prompt, specify Objectivity/Smalltalk for VisualWorks by typing its item number from the displayed list.

---

**NOTE** You must have a license for Objectivity/Smalltalk for VisualWorks to install it.

---

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 (see Table 6-1).
  - Runs the installation verification test `ooverify`, which checks that the release files are loaded properly. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 24.
7. Set up your image by following the instructions in “Setting Up VisualWorks” on page 48 or “Setting Up VisualWorks With ENVY/Developer” on page 49.
  8. If you intend to store Objectivity/Smalltalk for VisualWorks persistent collections in federated databases created prior to Release 5.2, make sure you have upgraded the schemas of those federated databases (see “Upgrading Schemas” on page 23).
  9. Read:
    - The printed *Objectivity Release Notes* for new and changed features.
    - The online release notes 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 setup program installs various files in subdirectories of the *installDir/arch* directory, as shown in Table 6-2.

**Table 6-2:** Objectivity/Smalltalk for VisualWorks Release Files in *installDir/arch*

Subdirectory	File	Description
bin	oogc	Garbage collection utility
doc	smalltalk.pdf	PDF file for the <i>Objectivity/Smalltalk for VisualWorks</i> online book (see “Viewing Online Books” on page 12)
lib	<i>Filenames differ on each platform</i>	Objectivity/Smalltalk shared library (bridge to Objectivity/DB kernel)

**Table 6-2:** Objectivity/Smalltalk for VisualWorks Release Files in *installDir/arch*

Subdirectory	File	Description
etc/smalltlk	objyDB.pcl	External interface class required for developing and deploying applications in VisualWorks
	objyDB.pst	Parcel source file for Objectivity/Smalltalk for VisualWorks
	objyDB.st	Objectivity/Smalltalk file-in 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 the Objectivity/Smalltalk for VisualWorks book for details.
	objyFTO.st <i>or</i> objyFTO.dat	Objectivity/DB Fault Tolerant Option file-in
	objyDRO.st <i>or</i> objyDRO.dat	Objectivity/DB Data Replication Option file-in

## Setting Up VisualWorks

This section describes how to set up VisualWorks for use with Objectivity/Smalltalk. (If you use VisualWorks with ENVY/Developer, go to the next section instead.)

1. Copy the file *installDir/arch/etc/smalltlk/objyDB.pcl* into the directory that contains your VisualWorks image.
2. Start VisualWorks.
3. File in the file *installDir/arch/etc/smalltlk/objyDB.st*.
4. Click **OK** on the prompt **Please provide a fully qualified filename:** to allow the installation to complete.
5. (Optional) If you purchased the Objectivity/DB Fault Tolerant Option, file in the file *objyFTO.st*.
6. (Optional) If you purchased the Objectivity/DB Data Replication Option, file in the file *objyDRO.st*.
7. Save your image.

8. (Optional) Delete the file `installDir/arch/etc/smalltalk/objyDB.dat` to free up disk space. This file is only used with ENVY/Developer.
9. **Important:** If you have any classes that define Objectivity/DB relationships, their generated relationship methods must be regenerated. To do this:
  - a. Install Objectivity/Smalltalk for VisualWorks into a fresh image.
  - b. File in your development code.
10. (Optional) Test the Objectivity/Smalltalk installation (see “Testing Objectivity/Smalltalk for VisualWorks Setup” on page 50).

## Setting Up VisualWorks With ENVY/Developer

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

1. Start VisualWorks with ENVY/Developer.
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. (Optional) If you purchased the Objectivity/DB Fault Tolerant Option, repeat step 3 for `objyFTO.dat`.
5. (Optional) If you purchased the Objectivity/DB Data Replication Option, repeat step 3 for `objyDRO.dat`.
6. Advise each Objectivity/Smalltalk user to modify his or her own VisualWorks image as follows:
  - a. Open a **Configuration Maps Browser**.
  - b. Use the option **load with required maps** for the configuration map **Objectivity/DB**.
  - c. Save the image.
7. **Important:** If you have any classes that define Objectivity/DB relationships, their generated relationship methods must be regenerated. To do this:
  - a. Make sure all such classes are loaded in the image.
  - b. Evaluate the following expression:
 

```
OoReleaseInstallUtility upgradeAllClasses
```
  - c. Repeat step 7b as many times as is necessary (that is, if not all of the class definitions are available). You may also run this method for individual classes by evaluating the following expression:
 

```
OoReleaseInstallUtility upgradeClass: aClass
```

8. (Optional) Test the product installation on ENVY/Developer (see “Testing Objectivity/Smalltalk for VisualWorks Setup” below).
9. (Optional) After importing the configuration maps and sample application code (see the Objectivity/Smalltalk for VisualWorks book) 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 by evaluating the expression:

```
OoReleaseInstallUtility>>verifyInstall
```

This method sends output to the Transcript.

You can test the basic functionality of Objectivity/Smalltalk for VisualWorks by evaluating:

```
OoReleaseInstallUtility>>verifyInstall: bootFilePath
```

where

*bootFilePath* Path to the boot file of the federated database

## Objectivity/SQL++ Installation

---

This chapter describes the requirements and steps for installing Objectivity/SQL++ on a UNIX platform. Objectivity/SQL++ provides ANSI-standard SQL access to Objectivity/DB, with object-oriented extensions to SQL. Objectivity/SQL++ has three components:

- The Objectivity/SQL++ ODBC server, a process that enables ODBC-compliant applications to access Objectivity/DB databases. (These applications must be configured to use the separately installed Objectivity/SQL++ ODBC Driver.)
- Interactive SQL++, a tool for interactively submitting SQL statements or scripts to an Objectivity/DB database.
- The Objectivity/SQL++ programming interface, which enables you to execute SQL statements from your C++ database applications.

## System Requirements

You can install Objectivity/SQL++ on the architectures shown in Table 7-1. Each architecture represents a combination of hardware and UNIX operating system.

---

**NOTE** See the online release notes on the Objectivity Technical Support web site for the currently supported operating-system versions. Contact Objectivity Customer Support to get access to this web site.

---

**Table 7-1:** Supported UNIX Architectures

Hardware	Operating-System Version	Architecture Name
DEC Alpha	<i>See web site</i>	alphaosf1
HP 9000 Series 700/800	HP/UX 10.20 HP/UX 11.0	hp9000s700 hprisc

**Table 7-1:** Supported UNIX Architectures (Continued)

Hardware	Operating-System Version	Architecture Name
IBM RISC System/6000	See web site	ibmrs6000
Silicon Graphics IRIS	See web site	iris
SPARCstation	2.6 Solaris 7	solaris4 solaris7

## Software Requirements

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

- Objectivity/DB (see Chapter 1)
- The vendor-supplied C++ compiler for your architecture

## Hardware Requirements

Your workstation must have a CD-ROM drive that is either local or accessible on the network.

## Installing Objectivity/SQL++

To install Objectivity/SQL++:

1. Log in to your workstation as `root` (required only for mounting the CD-ROM).
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” above).
3. Insert the Objectivity distribution CD-ROM into the CD-ROM drive and make sure the CD-ROM is mounted. See your operating system documentation for the proper mount command. On the following architectures, use the indicated option to suppress version numbers and case changes in filenames:
  - On the `alphaosf1` architecture, use the `-noversion` option.
  - On the `hp9000s700` and `hprisc` architectures, use the `-o cdcase` option.
4. Run the installation script provided on the distribution CD-ROM. Enter:
 

```
cdMountDir/install.csh
```

 where

`cdMountDir`      CD-ROM mount directory—for example, `/cdrom`

5. At the product prompt, specify Objectivity/SQL++ by typing its item number from the displayed list.

---

**NOTE** You must have a license for Objectivity/SQL++ to install it.

---

6. At the directory prompt, specify the directory in which Objectivity/SQL++ is installed (*sqlInstallDir*). You can choose the Objectivity/DB installation directory or another directory.

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 *sqlInstallDir/arch*, where *arch* is the architecture name for your platform (see Table 7-1).
  - Runs the installation verification test `ooverify`, which checks that the release files are loaded properly. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 24.
7. Create an account with the username *sysdba* and a password of your choice for the Objectivity/SQL++ database administrator. For more information, see the Objectivity/SQL++ book.
  8. Set the environment variable `OO_SQL_DIR` to be the path of the Objectivity/SQL++ installation directory. This environment variable is used for help and error messages. Enter:
 

```
setenv OO_SQL_DIR sqlInstallDir/arch
```
  9. Configure the network for the Objectivity/SQL++ ODBC server (see “Setting Up the Objectivity/SQL++ ODBC Server” on page 55).
  10. Test the Interactive SQL++ component of Objectivity/SQL++ by following the steps in “Testing Interactive SQL++” on page 56.
  11. Test the programming interface component of Objectivity/SQL++ by following the steps in “Testing the Programming Interface” on page 57.
  12. If you have installed the Objectivity/SQL++ ODBC Driver in the same network, you can set up the ODBC server component of Objectivity/SQL++ so you can test it with the driver. See:
    - “Preparing the ODBC Server for Testing” on page 58
    - The Objectivity/SQL++ ODBC Driver installation chapter in *Installation and Platform Notes for Windows*

**13. Read:**

- The printed *Objectivity Release Notes* for new and changed features.
- The online release notes 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 *sqlInstallDir/arch* directory, as shown in Table 7-2. For information about these files, see the Objectivity/SQL++ book.

**Table 7-2:** Objectivity/SQL++ Release Files in *sqlInstallDir/arch*

Subdirectory	Contains
bin	Executables for Interactive SQL++ and the Objectivity/SQL++ ODBC server
etc/sql	Subdirectories containing sample applications that use triggers or stored procedures
include	Include files for the Objectivity/SQL++ programming interface, triggers, and stored procedures
lib	Link libraries for user-created C++ applications that execute Objectivity/SQL++ statements
	Link libraries for rebuilding Interactive SQL++ or the ODBC server to accommodate user-defined triggers and stored procedures
demo/sql	Subdirectories containing applications for demonstration and testing (see “Testing Interactive SQL++” on page 56 and “Testing the Programming Interface” on page 57)

## Setting Up the Objectivity/SQL++ ODBC Server

You set up the Objectivity/SQL++ ODBC server by:

- Specifying its port number
- Setting up a boot file directory

### Specifying the ODBC Server's Port Number

You must identify the port number associated 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, `/etc/services`), or, if you are using the Network Information Service (NIS), add this line to the NIS services map:

```
oosqlnw      portNumber/tcp      # Objectivity/SQL++ Server
where
```

```
portNumber      TCP port number (a number greater than 1024)
```

**Important:** You must assign the *same* TCP port to the `oosqlnw` service on the Objectivity/SQL++ ODBC server host and on each Objectivity/SQL++ ODBC Driver host.

### Setting Up a Boot File Directory

You must enable the Objectivity/SQL++ ODBC server to locate the federated databases that are registered as data sources for ODBC-compliant client applications. Whenever a client application requires data from a registered federated database, the associated Objectivity/SQL++ ODBC Driver forwards the request to the ODBC server along with the simple name of the federated database's boot file. You must set up a directory where the ODBC server can find all such boot files. To set up a boot file directory for the Objectivity/SQL++ ODBC server:

1. Locate or create a directory that the Objectivity/SQL++ ODBC server can access through a locally understood name (a local pathname or an NFS network name).
2. For each federated database to be registered as a data source, copy the federated database's boot file into the directory you created in step 1. If necessary, you must rename the copy so that its name does not exceed 10 characters (including any filename extension) or contain spaces.

3. Set the `OO_FDDB_PATH` environment variable to be the path for the directory containing the boot files—for example, enter:
 

```
setenv OO_FDDB_PATH /usr/oodata
```
4. Each time a new federated database is registered as a data source, copy its boot file into the boot file directory.

## 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 demo application. This demo application builds a sample Objectivity/DB database that is then queried through Interactive SQL++.

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

1. Copy the Interactive SQL++ demo directory to a new location and change your working directory to this location. For example, enter:
 

```
cp -r sqlInstallDir/arch/demo/sql/ooisql /usr/ooisql_demo
cd /usr/ooisql_demo
```
2. Edit the makefile in the directory you just created (in this example, `/usr/ooisql_demo`) to set variables as appropriate to your installation. The lines containing these variables are near the beginning of the file:
  - Set `INSTALL_DIR` to the location of the Objectivity/DB installation directory—for example:
 

```
INSTALL_DIR = /usr/object
```
  - Set `SQL_ROOT` to the location of the Objectivity/SQL++ installation directory `sqlInstallDir` (which may be different from `INSTALL_DIR`)—for example:
 

```
SQL_ROOT = /usr/object_sql
```
  - If your lock server runs on a remote host, set `LS_HOST` to the name of the lock server host—for example:
 

```
LS_HOST = myLockServerHost
```

 If your lock server runs locally, leave the default value for `LS_HOST`.
3. Edit the demo file to set `passwd` to be the password of the Objectivity/SQL++ administrator account (`sysstpe`), which you created in step 7 on page 53—for example:
 

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

```
make
```

**6. Run the demo 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.
```

## Testing the Programming Interface

You can test whether the Objectivity/SQL++ programming interface is set up correctly by building and running the provided demo application. The demo application is a C++ application that uses the Objectivity/SQL++ interface to query and modify a federated database. You can also inspect the demo application to see how to use various Objectivity/SQL++ programming interface features.

To build and run the demo application for the Objectivity/SQL++ programming interface:

1. Prepare the federated database from the Interactive SQL++ demo for reuse in this demo:
  - If you have *not* already run the Interactive SQL++ demo application, perform steps 1 through 5 on page 56 (*do not perform step 6*).
  - If you *have* already run the Interactive SQL++ demo application:
    - Change to the demo directory you used (for example, `/usr/ooisql_demo`).
    - Check whether the lock server is running; start it, if necessary.
    - Clean up the directory by entering `make clean`
    - Re-create the federated database and application by entering `make`
2. Copy the Objectivity/SQL++ interface demo directory to a new location and change your working directory to this location. For example, enter:
 

```
cp -r sqlInstallDir/arch/demo/sql/ooapi /usr/ooapi_demo
cd /usr/ooapi_demo
```
3. Edit the makefile in the directory you just created (in this example, `/usr/ooapi_demo`) to set variables as appropriate to your installation. The lines containing these variables are near the beginning of the file.
  - Set `INSTALL_DIR` to the location of the Objectivity/DB installation directory—for example:
 

```
INSTALL_DIR = /usr/object
```

- Set `SQL_ROOT` to the location of the Objectivity/SQL++ installation directory `sqlInstallDir` (which may be different from `INSTALL_DIR`)—for example:  
`SQL_ROOT = /usr/object_sql`
4. Edit the demo file:
    - a. Set the value of `ooisqldemo` to the location of the Interactive SQL++ demo that you built in step 1—for example:  
`ooisqldemo=/usr/ooisql_demo`
    - b. Set the value of `passwd` to the password of the Objectivity/SQL++ administrator account (`sysstpe`), which you created on page 53—for example:  
`passwd=adminPassword`
  5. Build the demo application. Enter:  
`make`
  6. Run the demo application. Enter:  
`./demo`

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

```
Running the Objectivity/SQL++ Programming Interface test.
Comparing the results.
Test PASSED -- The expected results were achieved.
```

## Preparing the ODBC Server for Testing

You can test whether the Objectivity/SQL++ ODBC server can work with an installed Objectivity/SQL++ ODBC Driver by:

- Following the steps in this section to prepare the demo federated database and ODBC server
- Browsing the demo federated database through the Objectivity/SQL++ ODBC Driver (see the “Objectivity/SQL++ ODBC Driver Installation” of *Installation and Platform Notes for Windows*)

At some sites, a database administrator or a system administrator may be responsible for installing Objectivity/SQL++ and 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.

Perform the following steps on the Objectivity/SQL++ ODBC server host to prepare for the Objectivity/SQL++ ODBC Driver demo:

1. If you have not already done so, run the Interactive SQL++ demo (steps 1 through 6 on page 56) to create and populate the demo federated database to be browsed. Be sure to leave the lock server running.
2. If you have not already done so, create the ODBC server's boot file directory and set its pathname as the value of the `OO_FDDB_PATH` environment variable (see "Setting Up a Boot File Directory" on page 55).
3. Copy the boot file `DEMO` from the Interactive SQL++ demo directory into the boot file directory. For example, enter:

```
cp /usr/isqldemo/DEMO /usr/oodata
```

4. Start the Objectivity/SQL++ ODBC server. At a command prompt, enter:  

```
oosqld start
```
5. 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 and the boot filename (`DEMO`).
6. 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 (`sysstpe`) will have access to these tables. To grant access rights to all users:

- a. Start Interactive SQL++ for the demo federated database. Type:

```
ooisql demo
```

- b. Enter the `TABLE` statement to obtain a list of the demo tables. Type:

```
table;
```

- c. For each table listed, grant all rights to every user with a login account on the Objectivity/SQL++ ODBC server host. Type 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;
```

The Objectivity/SQL++ ODBC Driver demo can now be performed repeatedly without requiring any cleanup of the federated database.



## Objectivity/FTO Installation

---

This chapter describes the requirements and steps for installing Objectivity/DB Fault Tolerant Option (Objectivity/FTO) on a UNIX platform. Objectivity/FTO enables you to separate an Objectivity/DB federated database into independent pieces called *autonomous partitions*. Objectivity/FTO distributes and relocates Objectivity/DB services so that each partition is self-sufficient in case a network or system failure occurs in another partition.

### System Requirements

You can install Objectivity/FTO on any of the UNIX architectures listed in Table 1-1 on page 11.

### Software Requirements

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

- Objectivity/DB (see Chapter 1)

### Hardware Requirements

Your workstation must have a CD-ROM drive that is either local or accessible on the network.

### Installing Objectivity/FTO

To install Objectivity/FTO:

1. Log in to your workstation as `root` (required only for mounting the CD-ROM).
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” above).

3. Insert the Objectivity distribution CD-ROM into the CD-ROM drive and make sure the CD-ROM is mounted. See your operating system documentation for the proper mount command. On the following architectures, use the indicated option to suppress version numbers and case changes in filenames:
  - On the `alphaosf1` architecture, use the `-noversion` option.
  - On the `hp9000s700` and `hprisc` architectures, use the `-o cdcase` option.
4. Run the installation script provided on the distribution CD-ROM.
  - On all architectures except `linux86`, enter:
 

```
cdMountDir/install.csh
```
  - On `linux86`, enter:
 

```
csh cdMountDir/install.csh
```

In these commands:

`cdMountDir`      CD-ROM mount directory—for example, `/cdrom`

5. At the product prompt, specify Objectivity/FTO by typing its item number from the displayed list.

---

**NOTE** You must have a license for Objectivity/FTO to install it.

---

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 (see Table 1-1 on page 11).
  - Runs the installation verification test `ooverify`, which checks that the release files are loaded properly. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 24.
7. If you also have Objectivity/Smalltalk for VisualWorks installed, follow the steps in “Setting Up Objectivity/Smalltalk for VisualWorks” on page 63.
8. Read:
  - The printed *Objectivity Release Notes* for new and changed features.
  - The online release notes 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/FTO

When you install Objectivity/FTO, its files are organized in subdirectories of the *installDir/arch* directory, as shown in Table 8-1:

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

Subdirectory	Contains
bin	Executables for Objectivity/FTO tools (see the Objectivity/FTO and Objectivity/DRO book).
doc	PDF file for the <i>Objectivity/FTO and Objectivity/DRO</i> online book (see “Viewing Online Books” on page 12).
etc/smalltlk	Files providing the Smalltalk programming interface to Objectivity/FTO.
lib	Object module for linking user-created C++ database applications with Objectivity/FTO (see “Linking User Applications With Objectivity/DB” on page 69).

## Setting Up Objectivity/Smalltalk for VisualWorks

To set up Objectivity/Smalltalk for VisualWorks to work with Objectivity/FTO:

1. Start VisualWorks, if necessary.
2. If you use VisualWorks without ENVY/Developer, file in:
 

```
installDir/arch/etc/smalltlk/objyFTO.st.
```
3. If you use VisualWorks with ENVY/Developer:
  - a. Open a **Configuration Maps Browser**.
  - b. Import the following configuration map into your ENVY server repository from *installDir/arch/etc/smalltlk/objyFTO.dat*.
 

**Objectivity/FTO**
  - c. Advise each Objectivity/Smalltalk for VisualWorks user to open a **Configuration Maps Browser** and use the **load with required maps** option for the configuration map **Objectivity/FTO**.
4. Save your image.



## Objectivity/DRO Installation

---

This chapter describes the requirements and steps for installing Objectivity/DB Data Replication Option (Objectivity/DRO) on a UNIX platform. Objectivity/DRO enables you to create and manage multiple copies of a database (called *database images*). Because each copy resides in a separate autonomous partition, if the network or system fails in one partition, you can access your data in another.

### System Requirements

You can install Objectivity/DRO on any of the UNIX architectures listed in Table 1-1 on page 11.

### Software Requirements

Objectivity/DRO 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 contain a replicated database (see “Setting Up the Advanced Multithreaded Server” on page 17).
- Objectivity/FTO (see Chapter 8)

### Hardware Requirements

Your workstation must have a CD-ROM drive that is either local or accessible on the network.

## Installing Objectivity/DRO

To install Objectivity/DRO:

1. Log in to your workstation as `root` (required only for mounting the CD-ROM).
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 65).
3. Insert the Objectivity distribution CD-ROM into the CD-ROM drive and make sure the CD-ROM is mounted. See your operating system documentation for the proper mount command. On the following architectures, use the indicated option to suppress version numbers and case changes in filenames:
  - On the `alphaosf1` architecture, use the `-noversion` option.
  - On the `hp9000s700` and `hprisc` architectures, use the `-o cdcase` option.
4. Run the installation script provided on the distribution CD-ROM.
  - On all architectures except `linux86`, enter:
 

```
cdMountDir/install.csh
```
  - On `linux86`, enter:
 

```
csh cdMountDir/install.csh
```

In these commands:

`cdMountDir`      CD-ROM mount directory—for example, `/cdrom`

5. At the product prompt, specify Objectivity/DRO by typing its item number from the displayed list.

---

**NOTE** You must have a license for Objectivity/DRO to install it.

---

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 (see Table 1-1 on page 11).
  - Runs the installation verification test `ooverify`, which checks that the release files are loaded properly. If `ooverify` displays error messages, see “Troubleshooting Installation” on page 24.
7. If you also have Objectivity/Smalltalk for VisualWorks installed, follow the steps in “Setting Up Objectivity/Smalltalk for VisualWorks” on page 67.

8. Verify that AMS is installed and configured on every host that is to contain a replicated database (see “Setting Up the Advanced Multithreaded Server” on page 17). Although AMS need not be running when you create an original database image, you must start AMS before you can create additional database images.
9. Read:
  - The printed *Objectivity Release Notes* for new and changed features.
  - The online release notes 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/DRO

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

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

Subdirectory	Contains
bin	Executables for Objectivity/DRO tools (see the Objectivity/FTO and Objectivity/DRO book)
doc	PDF file for the <i>Objectivity/FTO and Objectivity/DRO</i> online book (see “Viewing Online Books” on page 12).
etc/smalltlk	Files providing the Smalltalk programming interface to Objectivity/DRO
lib	Object module for linking user-created C++ database applications with Objectivity/DRO (see “Linking User Applications With Objectivity/DB” on page 69)

## Setting Up Objectivity/Smalltalk for VisualWorks

To set up Objectivity/Smalltalk for VisualWorks to work with Objectivity/DRO:

1. Start VisualWorks, if necessary.
2. If you use VisualWorks without ENVY/Developer, file in:
 

```
installDir/arch/etc/smalltlk/objyDRO.st.
```
3. If you use VisualWorks with ENVY/Developer:
  - a. Open a **Configuration Maps Browser**.



# 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 user applications to Objectivity/DB and other Objectivity products
- Using makefiles
- Application programming issues
- Debugging applications

## Linking User Applications With 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.
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 the Objectivity/DB debugger tools (see page 75). This library performs more runtime checking than liboo.a.

## 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.xx` 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, `alphaosf1`. See Table 1-1 on page 11 for a list of architecture names.

If your application uses persistent collections or the C++ programming interface of other Objectivity products, you must add other libraries or object modules to your link line (see “Linking to Additional Objectivity Products and Features” on page 73).

### Link Rules for Development Environments

Table A-2 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.xx*              Digits corresponding to the current Objectivity/DB release.

**Table A-2:** Shared Library Link Rules for Development

<code>alphaosf1</code>	<code>-Wl,-call_shared -LinstallDir/alphaosf1/lib -loox.xx -lpthread</code>
<code>hp9000s700</code>	<code>-LinstallDir/hp9000s700/lib -loox.xx</code>
<code>hprisc</code>	<code>-LinstallDir/hprisc/lib -loox.xx -lpthread</code>
<code>ibmrs6000</code>	<code>-LinstallDir/ibmrs6000/lib -loox.xx -brtl</code>
<code>iris</code>	<code>-n32 -mips3 -LinstallDir/iris/lib -loox.xx -lpthread</code>
<code>linux86</code>	<code>-LinstallDir/linux86/lib -loox -lrpcsvc -lnsl -ldl -lpthread</code>
<code>solaris4</code>	<code>-LinstallDir/solaris4/lib -loox.xx -mt</code>
<code>solaris7</code>	<code>-LinstallDir/solaris7/lib -loox.xx -mt -library=iostream,no%Cstd</code>

## Link Rules for End-User Environments

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

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

**Table A-3:** Shared Library Link Rules for Deployment

alphaosf1	-L <i>installDir</i> /alphaosf1/lib -l <i>oo.x.xx</i> -lpthread Use the -rpath linker option to specify <i>endUserInstallDir</i> .
hp9000s700	-L <i>installDir</i> /hp9000s700/lib -l <i>oo.x.xx</i> -Wl,+s -Wl,+b <i>endUserInstallDir</i> See the ld(1) man page for more information.
hprisc	-L <i>installDir</i> /hprisc/lib -l <i>oo.x.xx</i> -Wl,+s -Wl,+b <i>endUserInstallDir</i> -lpthread See the ld(1) man page for more information.
ibmrs6000	-L <i>installDir</i> /ibmrs6000/lib -L <i>endUserInstallDir</i> -l <i>oo.x.xx</i> -brtl The two -L arguments specify the directories to be searched for Objectivity/DB libraries at both link time and runtime. The loader searches both of these paths at runtime for dynamic loading of the Objectivity/DB shared library. If these paths are the same—for example, /usr/object/lib—only one -L argument is needed.
iris	-n32 -mips3 -L <i>installDir</i> /iris/lib -l <i>oo.x.xx</i> -lpthread Use the -rpath linker option to specify <i>endUserInstallDir</i> .
linux86	-L <i>installDir</i> /linux86/lib -l <i>oo</i> -lrpcsvc -lnsl -ldl -lpthread
solaris4	-L <i>installDir</i> /solaris4/lib -l <i>oo.x.xx</i> -mt Define LD_RUN_PATH prior to compiling and linking.
solaris7	-L <i>installDir</i> /solaris7/lib -l <i>oo.x.xx</i> -mt -library=iostream,no%Cstd 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;
```

## Linking Under AIX

### Linking to Required System Libraries

Objectivity/DB must link to several multithreading system libraries. The easiest way to add these libraries to your link line is to use the multithreaded version of the AIX C++ compiler (`x1C_r`) instead of the normal version (`x1C`) when linking your application. For example:

```
x1C_r -o program a.o b.o c.o
      -LinstallDir/ibmrs6000/lib -lOO.x.xx -brtl
```

If you omit the `-brtl` flag, your application may link successfully, but result in a core dump when you try to run it.

### Id: 0711-317 Linking Errors

You may get the following errors when linking to Objectivity/DB libraries under AIX:

```
ld: 0711-317 ERROR: Undefined symbol: SOMClassClassData
ld: 0711-317 ERROR: Undefined symbol: SOMObjectClassData
```

To work around this problem, use the following link flag:

```
-bI:/usr/lpp/x1C/lib/libC.imp
```

### Overloaded Functions Error

You may get the following overloaded functions error when linking to Objectivity/DB libraries under AIX:

```
"./unistd.h", line 44: error: cannot overload functions
distinguished by return type alone char *sbrk(int);
```

To work around this problem, modify the system header file located in `/usr/lpp/x1C/include/unistd.h`. Change line 44 from:

```
char *sbrk(int);
```

to:

```
#ifndef OO_DDL_TRANSLATION
char *sbrk(int);
#endif
```

## Dynamic C++ Library Needed for Deployment on AIX

Objectivity/DB tools are linked to the AIX dynamic C++ library. Consequently, end users on IBM RISC System/6000 workstations must install this library on their machines to run Objectivity/DB tools. Instruct end users to install the `xlc.rte` component supplied on their AIX 4.1 CD-ROM.

## Linking to Additional Objectivity Products and Features

If your application uses the Objectivity/C++ persistent collections feature or the C++ programming interfaces of other Objectivity products, you must augment the link rules given in Table A-2 and Table A-3. For each product or feature used, add the object module or library indicated in Table A-4 to your link line before `liboo.a` or the shared Objectivity/DB library.

---

**NOTE** Shared library names vary on different architectures and contain digits `x.xx` corresponding to the current Objectivity/DB release.

---

**Table A-4:** Other Objectivity Object Modules and Libraries

To Use	Link to	
Objectivity/DRO and Objectivity/FTO	<code>ooRepl.o</code>	
Objectivity/FTO <i>only</i>	<code>ooPart.o</code>	
Objectivity/SQL++	<code>-l<code>ooakit.x.xx</code></code> <code>lib<code>ooakit.a</code></code>	On the <code>iris</code> architecture On all other architectures
Objectivity/C++ persistent collections	<code>-l<code>oo_co</code></code> <code>-l<code>oo_co.x.xx</code></code>	On the <code>linux86</code> architecture On all other architectures  Link with the Objectivity/DB shared library (not <code>liboo.a</code> ).
Objectivity/C++ Active Schema	<code>-l<code>oo_as</code></code> <code>-l<code>oo_as.x.xx</code></code>	On the <code>linux86</code> architecture On all other architectures  Link with the Objectivity/DB shared library (not <code>liboo.a</code> ).

**Table A-4:** Other Objectivity Object Modules and Libraries (Continued)

To Use	Link to
Objectivity/C++ STL	<div style="display: flex; justify-content: space-between;"> <div style="width: 45%;"> <p>libobjystl.a</p> <p>libospace.a</p> </div> <div style="width: 50%;"> <p>Objectivity/C++ STL library</p> <p>ObjectSpace STL library</p> </div> </div> <p>Link to <i>both</i> libraries. To do this, include the <code>local.cfg</code> file in your makefile, then add <code>libobjystl.a</code> and the <code>OLIBS</code> variable to your link line (see “Using Makefiles” on page 74).</p>

## Using Makefiles

### Objectivity/C++ Applications

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. Use the makefile for the C++ demo applications as a template for your own makefile (see “Testing Objectivity/C++ Setup” on page 30). The sample makefile is in:

```
installDir/arch/demo/CC/Makefile
```

### Objectivity/C++ STL Applications

For Objectivity/C++ STL applications, you can use the sample makefile in:

```
installDir/arch/ToolKit/ospace/stl/d_examples
```

If you want to use a makefile of your own, this makefile should include the following configuration file:

```
installDir/arch/ToolKit/config/local.cfg
```

You can then use the compile flags and link flags (such as `OLIBS`) defined in this file.

## Application Programming Issues

### Signal Handling

The Objectivity/DB predefined signal handler catches the following UNIX signals: `SIGINT`, `SIGQUIT`, `SIGILL`, `SIGABRT`, `SIGFPE`, `SIGBUS`, `SIGSEGV`, `SIGHUP`, `SIGPIPE`, `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 from the default value of 64 to a higher limit, such as 128. The **csh** and **ksh** commands to increase the limit are shown below. If you still experience errors, you should increase the limit incrementally.

---

**EXAMPLE** **csh:**

```
limit descriptors 128
```

**ksh:**

```
ulimit -n 128
```

---

## 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 the Objectivity/DB administration book).
- Run your application in debug mode for data verification and event tracing (see the Objectivity/C++ book).

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 an object handle. Doing so displays the object identifier (OID) of the persistent object that the handle references.

---

**EXAMPLE** Assume your application declares a handle variable `oopvar` to a persistent object. To print this variable, you enter the following `dbx` debugger command:

```
print oopvar
```

This command prints the following representation of the OID 2-2-25-144, which is the federated database address of the persistent object pointed to by `oopvar`:

```
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 the Objectivity/DB administration book.

## 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 guide.

This appendix provides information about:

- Running an Objectivity for Java application
- Setting the file descriptor limit
- Memory requirements

### Running an Objectivity for Java Application

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.

When you run an Objectivity for Java application, you should set the stack size to 1 megabyte or more. An inadequate stack size may cause an Objectivity for Java application to terminate with a segmentation violation for no apparent reason.

#### Changing the Stack Size

You change the default thread stack size for the Java Virtual Machine with the option:

```
-sssize
```

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 -ss2m classname
```

---

## 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 from the default value of 64 to a higher limit, such as 128. 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
```

**ks**h:

```
ulimit -n 128
```

---

## 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:

`-mssize` Sets the initial memory size.

`-mxsize` Sets the maximum memory size.

where

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

## 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.

### Lock Server Not Running

#### Solutions:

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

### Object Does Not Open

#### Solutions:

- Verify that the lock server process 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 process is running.
- If a `dbx` session was terminated while debugging an application, check if any locks remain.
- If your application is run in single-user mode (which turns off locking), 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 RPC 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

**Advanced Multithreaded Server (see AMS)**

**AIX linking issues** 72

**alphaosf1**

architecture 11, 41, 51

link rules 70, 71

**AMS**

setting up 17

using with Objectivity/DRO 65

**application**

compiling 74

linking 69

programming issues 74

**architectures, UNIX** 11

**autonomous partitions** 61

## C

**CLASSPATH environment variable** 43

**client host** 16

**compiling**

Objectivity/C++ application 74

Objectivity/C++ STL application 74

**configuration file** 35, 74

**customer support** 9

## D

**Data Definition Language (see DDL)**

**data packet size, used with NFS** 17

**data server**

host 16

software 16

**database format**

compatibility among Objectivity/DB

releases 20

determining 20

upgrading Release 3.x 20

upgrading Release 4.0.2 21

**DDL**

files 27

processor 27

**debugging**

compiling and linking for 76

printing

object handles 76

objects 76

**demo applications**

Interactive SQL++ 56

Objectivity for Java 44

Objectivity/C++ 30

Objectivity/C++ STL 35

Objectivity/SQL++ programming  
interface 57

**documentation (see online books)**

## E

**environment variables**

CLASSPATH 43

LD\_LIBRARY\_PATH 14

LD\_LIBRARYN32\_PATH 14

LIBPATH 14

## F

- OO\_FD\_BOOT 20, 79
- OO\_FDDB\_PATH 56
- OO\_NFS\_MAX\_DATA 17, 80
- OO\_RPC\_TIMEOUT 80
- OO\_SQL\_DIR 53
- PATH 13
- SHLIB\_PATH 14
- THREADS\_FLAG 43
- XBMLANGPATH 19
- XFILESEARCHPATH 19

### ENVY/Developer

- requirements 46
- setting up 49

### errors

- lock server 25
- running an application 79
- verifying installation 24

## F

### federated database

- finding out database format 20
- upgrading to the current release 20

### file descriptor limit

- specifying for Java 75, 78

## H

### hp9000s700

- architecture 11, 51
- link rules 70, 71

### hprisc

- architecture 11, 41, 45, 51
- link rules 70, 71

## I

### ibmrs6000

- architecture 11, 41, 52
- link rules 70, 71
- linking issues 72

### index

- upgrading Release 5.0 23

### installation, troubleshooting 24

### installing

- Objectivity for Java 41
- Objectivity/AS 37
- Objectivity/C++ 27
- Objectivity/C++ STL 33
- Objectivity/DB 11
- Objectivity/DDL 27
- Objectivity/DRO 65
- Objectivity/FTO 61
- Objectivity/Smalltalk for VisualWorks 45
- Objectivity/SQL++ 51

### Interactive SQL++

- defined 51
- demo applications 56
- testing 56

### iris

- architecture 11, 41, 52
- link rules 70, 71

## L

### LD\_LIBRARY\_PATH environment variable 14

### LD\_LIBRARYN32\_PATH environment variable 14

### libobjstl.a 74

### liboo.a 69

### liboo\_adm.a 69

### liboo\_dbx.a 69

### libooakit.a 73

### libospace.a 74

### LIBPATH environment variable 14

### library

- administration 69
- debug 69
- Objectivity/AS 73
- Objectivity/C++ persistent collections 73
- Objectivity/C++ STL 74
- Objectivity/DB
  - shared runtime 70
  - static runtime 69
- Objectivity/SQL++ 73
- reentrant C 74
- standard 69

**linking Objectivity/C++ application 69**

- link rules
  - development environments 70
  - end-user deployment 71
- under AIX 72
- with Objectivity/AS 73
- with Objectivity/C++ STL 74
- with Objectivity/DRO 73
- with Objectivity/FTO 73
- with Objectivity/SQL++ 73
- with persistent collections 73
- with Purify 72

**linux86**

- architecture 11, 41, 45
- link rules 70, 71

**local.cfg configuration file 35, 74****lock server 15**

- compatibility among Objectivity/DB releases 19
- errors 25
- port 16
- setting up 15
- system directory 15

**M****makefile**

- configuring for Objectivity/C++ STL 35, 74
- Interactive SQL++ demo 56
- Objectivity/C++ demo application 30, 74
- Objectivity/C++ STL demo application 36, 74
- Objectivity/SQL++ programming interface demo 57

**memory**

- increasing for Java 78

**N****Network File System (see NFS)****NFS**

- data packet size 17, 80
- setting up 16

**O****object module**

- Objectivity/DRO 73
- Objectivity/FTO 73

**Objectivity for Java**

- compiler requirements 42
- demo applications 44
- increasing memory 78
- installing 41
- release files 44
- specifying file descriptor limit 75, 78
- specifying stack size 77
- system requirements 41
- testing 44
- upgrading 4.0.10 federated database 44

**Objectivity servers**

- AMS 16
- lock server 15

**Objectivity/AS**

- installing 37
- library 73
- release files 39
- system requirements 37

**Objectivity/C++**

- compiler requirements 27
- compiling 74
- debugging application 75
- demo applications 30
- installing 27
- linking application 69
- persistent collections library 73
- programming issues 74
- release files 29
- system requirements 27
- testing 30

**Objectivity/C++ Active Schema**

(see Objectivity/AS)

**Objectivity/C++ Standard Template Library**

(see Objectivity/C++ STL)

**Objectivity/C++ STL**

- compiling 74
- configuration file 35, 74
- demo applications 35

- installing 33
- library 74
- release files 35
- system requirements 33
- testing 35
- Objectivity/DB**
  - graphical tools 17
  - installing 11
  - release files 15
  - shared runtime library 70
  - static runtime library 69
  - system requirements 11
  - upgrading existing federated databases 20
- Objectivity/DDL**
  - installing 27
  - release files 29
  - system requirements 27
  - testing 30
- Objectivity/DRO**
  - C++ object module 73
  - installing 65
  - release files 67
  - system requirements 65
- Objectivity/FTO**
  - C++ object module 73
  - installing 61
  - release files 63
  - system requirements 61
- Objectivity/Smalltalk for VisualWorks**
  - installing 45
  - release files 47
  - setup for Objectivity/DRO 67
  - setup for Objectivity/FTO 63
  - system requirements 45
  - testing 50
- Objectivity/SQL++**
  - C++ library 73
  - demo applications 57
  - installing 51
  - Interactive SQL++
    - defined 51
    - testing 56
  - ODBC server
    - defined 51
    - setting up 55
    - TCP/IP port 55
    - testing 58
  - programming interface
    - defined 51
    - testing 57
  - release files 54
  - system requirements 51
  - testing
    - Interactive SQL++ 56
    - ODBC server 58
    - programming interface 57
- ObjectSpace STL** 33
  - library 74
- ODBC server (see Objectivity/SQL++ ODBC server)**
- online books**
  - location 15
  - viewing 12
- OO\_FD\_BOOT environment variable** 20, 79
- OO\_FDDB\_PATH environment variable** 56
- OO\_NFS\_MAX\_DATA environment variable** 17, 80
- OO\_RPC\_TIMEOUT environment variable** 80
- OO\_SQL\_DIR environment variable** 53
- ooconfig script** 29
- ooddllx** 29
- oofile tool** 20
- oolockserver** 16
- ooPart.o** 73
- ooRepl.o** 73
- ooschemaupgrade tool** 23
- oostartams** 17
- ooupgrade tool** 21
- ooupgrade402 tool** 22
- ooverify**
  - errors 24
  - testing installation 13

**P****packet size, used with NFS** 17**PATH environment variable** 13**persistent collections**

linking Objectivity/C++ applications 73

upgrading schema for 23

**predefined signal handler** 74**R****reentrant C libraries** 74**RPC timeout period** 80**running a multithreaded Java application** 75**S****sample applications**

Interactive SQL++ 56

Objectivity for Java 44

Objectivity/C++ 30

Objectivity/C++ STL 35

Objectivity/SQL++ 57

**schema**

compatibility among Objectivity/DB

releases 23

upgrading for persistent collections 23

**setting up**

AMS 17

lock server 15

NFS 16

Objectivity/DB graphical tools 17

Objectivity/SQL++ ODBC server 55

VisualWorks 48

VisualWorks with ENVY/Developer 49

**SHLIB\_PATH environment variable** 14**signal handler, predefined** 74**signals** 74**solaris4**

architecture 11, 41, 45, 52

link rules 70, 71

**solaris7**

architecture 11, 41, 45, 52

link rules 70, 71

**SPARCstation (see solaris4)****stack size**

specifying for C++ 75

specifying for Java 77

**Standard Template Library**

(see Objectivity/C++ STL)

**T****THREADS\_FLAG environment variable** 43**troubleshooting**

applications 79

installation 24

**U****UNIX-specific development issues**

Objectivity for Java 77

Objectivity/C++ 69

**upgrading existing federated databases** 20**V****VisualWorks**

requirements 46

setting up 48

**X****X Window System** 12

Objectivity/DB graphical tools 17

**XBMLANGPATH environment variable** 19**XFILESEARCHPATH environment variable**

19

