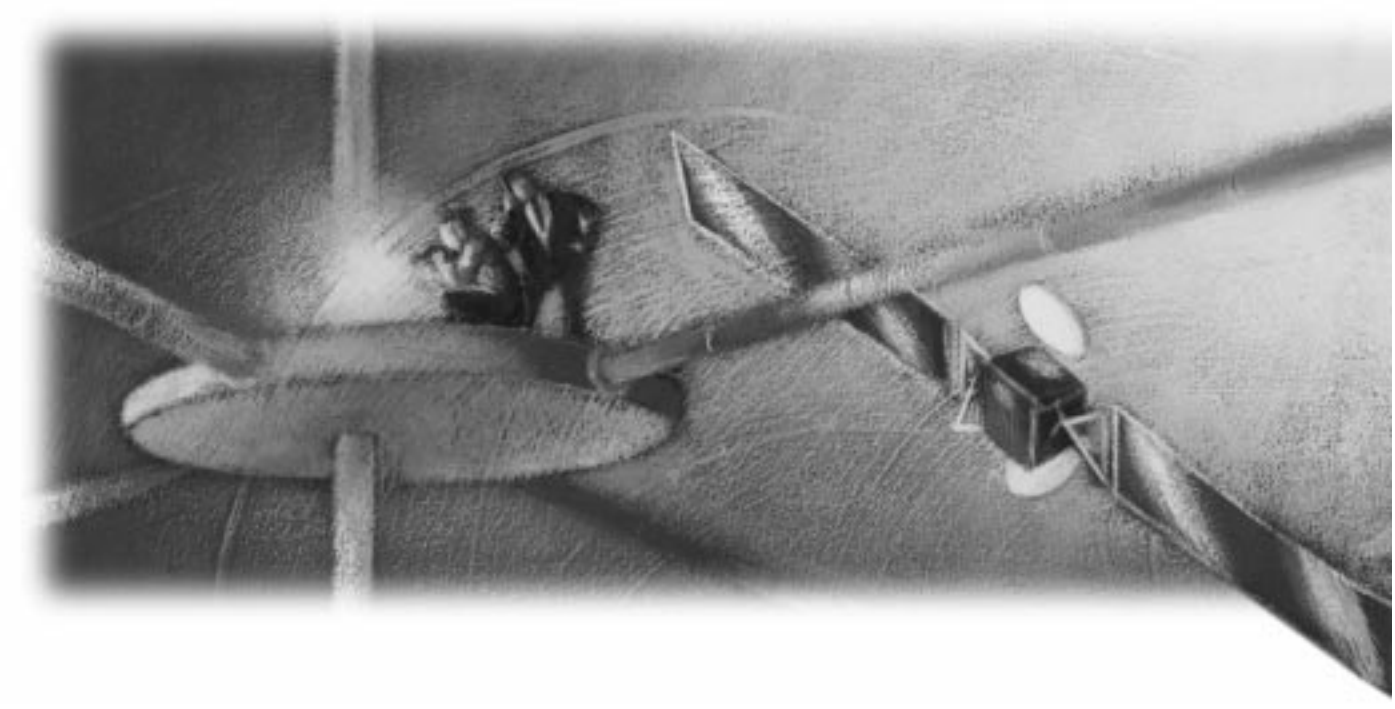

Installation and Platform Notes for UNIX

Release 6.0

Installation and Platform Notes for UNIX

Part Number: 60-IUNX-0

Release 6.0, October 19, 2000

Contents

About This Book

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

Audience

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

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

Organization

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

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

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

Conventions and Abbreviations

Navigation

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

Typographical Conventions

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

Abbreviations

(administration)	Feature intended for database administration tasks
(*FTO*)	Feature of the Objectivity/DB Fault Tolerant Option product
(*DRO*)	Feature of the Objectivity/DB Data Replication Option product
(*IPLS*)	Feature of the Objectivity/DB In-Process Lock Server Option product
(*ODMG*)	Feature conforming to the Object Database Management Group interface

Command Syntax Symbols

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

Command and Code Conventions

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

Getting Help

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

Technical Support Web Site

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

How to Reach Objectivity Customer Support

You can contact Objectivity Customer Support by:

- **Telephone:** Call 1.650.254.7100 *or* 1.800.SOS.OBJY (1.800.767.6259) Monday through Friday between 6:00 A.M. and 6:00 P.M. Pacific Time, and ask for Customer Support.

  The toll-free 800 number can be dialed *only* within the 48 contiguous states of the United States and Canada.
- **Fax:** Send a fax to Objectivity at 1.650.254.7171.
- **Electronic Mail:** Send electronic mail to *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, 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

1

Objectivity/DB Installation

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

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

System Requirements

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

NOTE See the 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	*See web site*	`alphaosf1`
HP 9000 Series 700/800	*See web site*	`hprisc`
IBM RISC System/6000	*See web site*	`ibmrs6000`

Table 1-1: Supported UNIX Architectures (Continued)

Hardware	Operating System Version	Architecture Name
Intel Pentium or greater	*See web site*	`linux86`
Silicon Graphics IRIS	*See web site*	`iris`
SPARCstation	Solaris 2.6 Solaris 7 and Solaris 8	`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)

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 Acrobat Reader from Adobe's online services. Use your World Wide Web browser to access the web site `www.adobe.com`.

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`, if required for mounting a CD.
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 in this book).
3. Locate or create an installation directory for Objectivity / DB. To create the installation directory, enter:

   `mkdir` *`installDir`*

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

4. Give all Objectivity / DB users both read and execute permission to *installDir*.
5. Place the Objectivity distribution CD in the CD-ROM drive and make sure the CD 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.
6. Run the installation script provided on the distribution CD. Enter:
   cdMountDir`/install.csh`
   where *cdMountDir* is the CD mount directory—for example, `/cdrom`.
7. At the product prompt, specify the products 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 every product you 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 23.
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 `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 16).
12. Set up data server software (NFS or AMS) to provide remote data access (see "Setting Up Data Server Software" on page 17).
13. Set up Objectivity / DB administration and database tools (see "Setting Up Objectivity / DB Graphical Tools" on page 19).
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 20. You may need to upgrade the indexes or schemas of such federated databases before you can access them.
15. If you plan to continue using tools or applications built with earlier Objectivity / DB releases, read "Maintaining Older Objectivity / DB Releases" on page 22. You may need to change the host on which you run an older version of the lock server.
16. If you specified multiple products in step 7, read the installation chapters for those products, and follow any additional steps for setting them up.
17. Familiarize yourself with Objectivity online books. To do so, start Acrobat Reader and display the file *`installDir`*`/doc/ObjyBooks.pdf`. A PDF file is displayed containing links to the online books.

NOTE The online books installed with Objectivity / DB include *Objectivity Release Notes*, *Objectivity/DB Administration*, *Monitoring Lock Server Performance*, and *Installation and Platform Notes*. The books for other Objectivity products are provided when you install those products.

---

18. Read:
    - The *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the PDF file *`installDir`*`/doc/ReleaseNotes.pdf`.
    - The 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 18).
doc	PDF files for the online books *Objectivity Release Notes*, *Installation and Platform Notes*, *Objectivity/DB Administration*, and *Monitoring Lock-Server Performance*. PDF file for *Objectivity Books* (links to the online books).
etc	Graphics support for Objectivity/DB tools.
	Upgrade file for adding persistent-collection types to pre-Release 5.2 federated database schemas.
include	Include file for the lock-server performance-monitoring interface.
lib	Shared libraries for Objectivity/DB tools and the Objectivity/DB kernel.
	Static and shared libraries for linking custom lock-server performance-monitoring programs (see “Linking a Lock-Server Performance-Monitoring Program” on page 76).

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

For security purposes, you should consider running the lock server under a special-purpose user account in a group that can be granted just the minimum necessary permissions. These permissions are described in Chapter 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 when you start the lock server on a particular workstation for the first time:

1. Log in as `root` on the workstation that is to run the lock server.
2. Create the Objectivity 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. To start the lock server under the user account *`userName`*, you can enter a command such as the following:
   ```
   su - userName -c "installDir/arch/bin/oolockserver"
   ```
5. If the lock server does not start, verify that you performed steps 2 and 3 correctly, and try starting the lock server again.

   The lock server will not start if another service already uses the TCP / IP port that the lock server expected to use. If you cannot reassign the other service to a different port, you may have to change the default port for the lock server. To do this, see Chapter 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, edit the workstation's startup script (usually `/etc/rc.local` or `/etc/init.d`) and add a command such as the one shown in step 4.

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

For security purposes, you should consider running AMS under a special-purpose user account in a group that can be granted just the minimum necessary permissions. These permissions are described in Chapter 8, "Advanced Multithreaded Server," in the Objectivity/DB administration book. When an application uses AMS, any database files created by the application are owned by the user account under which AMS was started.

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

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

1. Log in as `root` on the workstation that is to run AMS.
2. Verify that the workstation has an Objectivity server system 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. To start AMS under the user account *`userName`*, you can enter a command such as the following:

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

   AMS will not start if another service already uses the TCP/IP port that AMS expected to use. If you cannot reassign the other service to a different port, you 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, edit the workstation's startup script (usually `/etc/rc.local` or `/etc/init.d`) and add a command such as the one shown in step 3.

   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 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	In a Federated Database Created With	See Page
Indexes	Release 5.0	21
Schema	Any release prior to Release 5.2	21
(No upgrade required)	Release 5.2	—

After you have performed any necessary upgrades to an existing federated database, you can run tools and applications that are built with Objectivity / DB

Release 6.0. That is, you can access the existing federated database with newly developed applications or you can run existing applications that have been recompiled and relinked with the current Objectivity / DB release. Before you access a federated database using Release 6.0 tools and applications, however, you must ensure that its lock-server host is running the Release 6.0 lock server.

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 after Release 5.0.

➤ 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 6.0, 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.

NOTE You can skip this section for schemas created with Release 5.2.

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

```
ooschemaupgrade
    -infile installDir/arch/etc/ooCollectionsSchema.dmp
    bootFilePath
```

where *`bootFilePath`* is the 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.

Maintaining Older Objectivity/DB Releases

After installing Release 6.0 of Objectivity / DB and your chosen Objectivity programming interface, you can develop new applications or upgrade existing applications to take advantage of Release 6.0 features. (For information about upgrading existing applications, see *Objectivity Release Notes*.) As you develop Release 6.0 applications, you may also need to maintain deployed federated databases and applications that were built with an older Objectivity / DB release.

When setting up your development and maintenance environment you must decide whether to run a lock server from each release, and if so, where. You should take the following information into account.

Release 5.2 Lock Servers

Release 6.0 lock servers use a protocol that is compatible with the protocol used by Release 5.2 lock servers. Consequently, applications built with Release 5.2 can use a Release 6.0 lock server, so the same federated database can be accessed concurrently by both a Release 6.0 application and a Release 5.2 application.

Lock servers with compatible protocols use the same TCP / IP port, so you cannot run both a Release 6.0 lock server and a Release 5.2 lock server on the same computer at the same time. For this reason, you should replace the Release 5.2 lock server with a Release 6.0 lock server on each lock-server host.

Lock Servers Earlier than Release 5.2

Release 6.0 lock servers use a protocol that is incompatible with the protocol used by lock servers from any release prior to Release 5.2. Consequently, applications built with Release 5.1.*x* or earlier cannot use a Release 6.0 lock server, so you will have to keep the older lock server running.

The safest configuration is to run the Release 6.0 lock server and the older lock server on two different hosts. This may mean changing an existing federated database's lock-server host and starting the older lock server on that host; see Chapter 7, "Using a Lock Server," in the Objectivity / DB administration book.

Lock servers with incompatible protocols use different TCP / IP ports, so it is possible to run both a Release 6.0 lock server and a lock server from Release 5.1.x or earlier on the same computer. However, if a federated database specifies a lock-server host that is running multiple lock servers, you must guarantee that *all*

applications accessing a particular federated database have been built with the *same* release of Objectivity / DB (so they will all contact the same lock server).

WARNING Data corruption will occur if two applications contact different lock servers while accessing data in the same federated database.

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

ooverify: File or Directory Content Errors

Error messages produced by `ooverify`:

```
Files of the wrong size
Files with the wrong checksum
Required directories not listed in the manifest
Duplicate entries in the manifest file
Internal check
Bad format in manifest file, line...
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; 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. Rerun `ooverify` (enter *`installDir/arch/`*`bin/ooverify`). If the problem persists, your distribution CD 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.

NOTE You can safely ignore the error message `Wrong access mode`, if it is followed by a list of directories that all have the same permissions. The condition is caused by the `umask` setting of the `root` account that ran the installation script. You do not need to change permissions on any directories.

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. 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 16 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.

2

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 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.

Installing Objectivity/C++ or Objectivity/DDL

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

1. Log in to your workstation as `root`, if required for mounting a CD.
2. Verify that all required software has been completely and correctly installed (see "Software Requirements" on page 27).
3. Place the Objectivity distribution CD in the CD-ROM drive and make sure the CD 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` architectures, use the `-o cdcase` option.
4. Run the installation script provided on the distribution CD. Enter:

   `cdMountDir``/install.csh`

   where *`cdMountDir`* is the CD mount directory—for example, `/cdrom`.
5. At the product prompt, specify the products to be installed:
   - To install only 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 every product you install.

---

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

   The installation script:
   - Places the release files in subdirectories of *`installDir/arch`*, where *`arch`* is the architecture name for your platform (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 23.
   - 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 21).
9. Read:
   - Appendix A, "C++ Application Development," in this book for platform-specific information about using Objectivity / C++.
   - The *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the PDF file *installDir*`/doc/ReleaseNotes.pdf`.
   - The 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 (`ooconfig`) for configuring the DDL processor; executable (`ooddlx`) created by `ooconfig`.
`demo`	C++ database applications for demonstration and testing (see "Testing Objectivity/C++ Setup" on page 30).
`doc`	PDF files for the *Objectivity/C++ Data Definition Language*, *Objectivity/C++ Programmer's Guide*, and *Objectivity/C++ Programmer's Reference* online books.
`include`	Include files for the C++ programming interface.
`lib`	Static libraries for linking C++ applications (see "Linking Applications to Objectivity/DB" on page 71). Shared library for persistent collections (see "Linking to Additional Objectivity Products and Features" on page 75).

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 16)—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/oolockserver
   ```

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.

3

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 the UNIX architectures shown in Table 3-1. Each architecture represents a combination of hardware and UNIX operating system.

NOTE See the 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 3-1: Supported UNIX Architectures

Hardware	Operating System Version	Architecture Name
DEC Alpha	*See web site*	`alphaosf1`
HP 9000 Series 700/800	*See web site*	`hprisc`
IBM RISC System/6000	*See web site*	`ibmrs6000`
Intel Pentium or greater	*See web site*	`linux86`
SPARCstation	Solaris 2.6	`solaris4`

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)

Installing Objectivity/C++ STL

To install Objectivity / C++ STL:

1. Log in to your workstation as `root`, if required for mounting a CD.
2. Verify that all required software has been completely and correctly installed (see "Software Requirements" on page 34).
3. Place the Objectivity distribution CD in the CD-ROM drive and make sure the CD 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` architectures, use the `-o cdcase` option.
4. Run the installation script provided on the distribution CD. Enter:
   `cdMountDir``/install.csh`
   where *`cdMountDir`* is the CD 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 every product you install.

---

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

   The installation script:
   - 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 3-2).
   - 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 23.

7. (Optional) Test the installation of Objectivity / C++ STL by running the provided C++ demo applications (see "Testing Objectivity / C++ STL Setup" on page 36).
8. Read:
   - Appendix A, "C++ Application Development," in this book for platform-specific information about compiling and linking this product.
   - The *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the PDF file *installDir*/doc/ReleaseNotes.pdf.
   - The 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 3-2.

Table 3-2: Objectivity/C++ STL Release Files in *installDir/arch*/ToolKit

Subdirectory of ToolKit	Contains
config	Configuration file (local.cfg) to be included in makefiles (defines compile and link flags)
doc	ObjectSpace STL online documentation
lib	Static libraries for linking applications with ObjectSpace STL and Objectivity/C++ STL (see "Linking to Additional Objectivity Products and Features" on page 75)
ospace/stl	Include files and source files for ObjectSpace STL and Objectivity/C++ STL
ospace/stl/d_examples	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.

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 16)—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`
4. Start the lock server. Enter:
   `installDir/arch/bin/oolockserver`
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.

4

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)

Installing Objectivity/AS

To install Objectivity / AS:

1. Log in to your workstation as `root`, if required for mounting a CD.
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 37).
3. Place the Objectivity distribution CD in the CD-ROM drive and make sure the CD 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` architectures, use the `-o cdcase` option.

4. Run the installation script provided on the distribution CD. Enter:
   cdMountDir/install.csh
   where *cdMountDir* is the CD 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 every product you install.

---

6. At the directory prompt, specify the directory in which Objectivity / DB is installed (*installDir*). If you get an error message reporting insufficient disk space or incorrect permissions, press Enter to exit the installation script; fix the problem and restart the installation script (see step 4).
   The installation script:
   - Places the release files in subdirectories of *installDir/arch*, where *arch* is the architecture name for your platform (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 23.
7. Read:
   - Appendix A, "C++ Application Development," in this book for platform-specific information about compiling and linking this product.
   - The *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the PDF file *installDir*/doc/ReleaseNotes.pdf.
   - The 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 inTable 4-1.

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

Subdirectory	Contains
doc	PDF file for the *Objectivity/C++ Active Schema* online book
include	Include file ooas.h for the Objectivity/AS programming interface
lib	Shared link libraries for Objectivity/AS (see "Linking to Additional Objectivity Products and Features" on page 75)

5

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 1-1 on page 11.

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 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.
- A World Wide Web browser to view Objectivity for Java online books in HTML format.

Installing Objectivity for Java

To install Objectivity for Java:

1. Log in to your workstation as `root`, if required for mounting a CD.
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 39).

3. Place the Objectivity distribution CD in the CD-ROM drive and make sure the CD 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. Enter:
   `cdMountDir``/install.csh`
   where *`cdMountDir`* is the CD 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 every product you install.

---

6. At the directory prompt, specify the directory in which Objectivity / DB is installed (*`installDir`*). If you get an error message reporting insufficient disk space or incorrect permissions, press Enter to exit the installation script; then fix the problem and restart the installation script (see step 4).
   The installation script:
   - Places the release files in subdirectories of *`installDir/arch`*, where *`arch`* is the architecture name for your platform (see “Release Files for Objectivity for Java” on page 41).
   - 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 23.
7. Set the `CLASSPATH` environment variable as follows:
   `setenv CLASSPATH` *`installDir/arch`*`/java/lib/oojava.jar:`*`existingValues`*
   where *`existingValues`* represents existing class path components.
   For some application-development environments, you must 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 42).

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 21).
11. Read:
    - Appendix B, "Java Application Development," in this book for platform-specific information about using Objectivity for Java.
    - The *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the PDF file *installDir*/doc/ReleaseNotes.pdf.
    - The 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 5-1.

Table 5-1: Objectivity for Java Release Files in *installDir/arch*

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

a. View with Acrobat Reader.
b. View with an HTML browser.

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 5-1.

6

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 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	*See web site*	`hprisc`
IBM RISC System/6000	*See web site*	`ibmrs6000`
Intel Pentium or greater	*See web site*	`linux86`
SPARCstation	Solaris 2.6 Solaris 7 and Solaris 8	`solaris4` `solaris7`

Software Requirements

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

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

Note: See the release 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.

Installing Objectivity/Smalltalk for VisualWorks

To install Objectivity / Smalltalk for VisualWorks:

1. Log in to your workstation as `root`, if required for mounting a CD.
2. Verify that all required software has been completely and correctly installed (see "Software Requirements" on page 44).
3. Place the Objectivity distribution CD in the CD-ROM drive and make sure the CD is mounted. See your operating system documentation for the proper mount command. On the following architecture, 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. Enter:

   `cdMountDir``/install.csh`

   where *`cdMountDir`* is the CD 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 every product you install.

---

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

The installation script:

- Places the release files in subdirectories of *installDir/arch*, where *arch* is the architecture name for your platform (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 23.

7. Set up your image by following the instructions in "Setting Up VisualWorks" *or* "Setting Up VisualWorks With ENVY / Developer" on page 47.
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 21).
9. Read:
   - The *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the PDF file *installDir*`/doc/ReleaseNotes.pdf`.
   - The 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 Release Files in *installDir/arch*

Subdirectory	File	Description
`bin`	`oogc`	Garbage collection utility
`doc`	`smalltalk.pdf`	PDF file for the *Objectivity/Smalltalk for VisualWorks* online book
`lib`	*Filenames differ on each platform*	Objectivity/Smalltalk shared library (bridge to Objectivity/DB kernel)

Table 6-2: Objectivity/Smalltalk Release Files in *installDir/arch* (Continued)

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. *Do not* file in this file unless you plan to use the threadsafe option; see the Objectivity/Smalltalk for VisualWorks book for details.
	objyFTO.st *or* objyFTO.dat	Objectivity/DB Fault Tolerant Option file-in
	objyDRO.st *or* 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. Start VisualWorks with a fresh image.
2. File in the file:
   installDir/arch/etc/smalltlk/objyDB.st
3. Enter the following in the prompt **Please provide a fully qualified filename:**
   installDir/arch/etc/smalltlk/objyDB.pcl
4. Click **OK** 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 the image.

8. File in your development code.
9. (Optional) Delete the file *installDir/arch*/etc/smalltlk/objyDB.dat to free up disk space. This file is only used with ENVY / Developer.
10. (Optional) Test the Objectivity / Smalltalk installation (see "Testing Objectivity / Smalltalk for VisualWorks Setup" on page 47).

Setting Up VisualWorks With ENVY/Developer

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

1. Start VisualWorks with a fresh ENVY / Developer image.
2. Open a **Configuration Maps Browser**.
3. Import all of the configuration maps into your ENVY server repository from *installDir/arch*/etc/smalltlk/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. Use the option **load with required maps** for the configuration map **Objectivity/DB**.
7. Save the image.
8. File in your development code.
9. (Optional) Test the product installation on ENVY / Developer (see "Testing Objectivity / Smalltalk for VisualWorks Setup" below).
10. (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* is the path to the boot file of the federated database.

7

Objectivity/SQL++ Installation

This chapter describes the requirements and steps for installing Objectivity/SQL++ on a UNIX platform. Objectivity/SQL++ provides ANSI-standard SQL-92 access to 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. (Requires 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 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	*See web site*	`alphaosf1`
HP 9000 Series 700/800	*See web site*	`hprisc`

Table 7-1: Supported UNIX Architectures (Continued)

Hardware	Operating System Version	Architecture Name
Silicon Graphics IRIS	*See web site*	`iris`
SPARCstation	Solaris 2.6 Solaris 7 and Solaris 8	`solaris4` `solaris7`

Software Requirements

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

- Objectivity/DB (see Chapter 1)

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

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

Installing Objectivity/SQL++

To install Objectivity/SQL++:

1. Log in to your workstation as `root`, if required for mounting a CD.
2. Verify that all required software has been completely and correctly installed (see "Software Requirements" above).
3. Place the Objectivity distribution CD in the CD-ROM drive and make sure the CD 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` architectures, use the `-o cdcase` option.
4. Run the installation script provided on the distribution CD. Enter:
   `cdMountDir``/install.csh`
   where *`cdMountDir`* is the CD 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 every product you install.

---

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 23.

7. Create an account with the username `systpe` and a password of your choice. The Objectivity/SQL++ database administrator will use this account. 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 to locate help files and the files that map error codes to error messages. Enter:
   `setenv OO_SQL_DIR sqlInstallDir/arch`
9. Set up the Objectivity/SQL++ ODBC server by following the steps in "Setting Up the Objectivity/SQL++ ODBC Server" on page 52.
10. Test each Objectivity/SQL++ component you plan to use:
    - Test Interactive SQL++ by following the steps in "Testing Interactive SQL++" on page 54.
    - Test the programming interface by following the steps in "Testing the Programming Interface" on page 55.
    - Set up the ODBC server so you can test it together with an Objectivity/SQL++ ODBC Driver that has been installed in the same TCP/IP network. Follow the steps in "Preparing the ODBC Server for Testing" on page 56.
11. Read:
    - The *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the PDF file *installDir*`/doc/ReleaseNotes.pdf`.
    - The 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 demonstrate the use of triggers and stored procedures; subdirectory containing help files for Interactive SQL++
include	Include files for the Objectivity/SQL++ programming interface, triggers, and stored procedures
lib	Link libraries for C++ applications created with the Objectivity/SQL++ programming interface
	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 54 and “Testing the Programming Interface” on page 55)

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

1. Log in as `root`.
2. Add the following entry to the TCP/IP `services` file (typically, `/etc/services`):

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

If you are using the Network Information Service (NIS), you should ask your system administrator to perform the equivalent operation for your NIS configuration.

3. If another service already uses TCP/IP port 1990, either reassign that service to a different port (recommended) or assign a different port to the `oosqlnw` service.

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

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`
   - Set `LS_HOST` to the name of the lock server host—for example:
     `LS_HOST = myLockServerHost`
3. Edit the `demo` file to set `passwd` to be the password of the Objectivity / SQL++ administrator account (`systpe`), which you created in step 7 on page 51—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 54 (*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 (systpe), which you created on page 51—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

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

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

1. If you have not already done so, run the Interactive SQL++ demo (steps 1 through 6 on page 54) 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 53).
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 file name (`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 (`systpe`) 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
      ```

Users can now perform the Objectivity / SQL++ ODBC Driver demo described in the Objectivity / ODBC installation chapter of *Installation and Platform Notes for Windows* or *Objectivity/SQL++ ODBC Driver Guide*. The demo can be repeated as long as the lock server and the Objectivity / SQL++ ODBC server are both running.

8

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)

Installing Objectivity/FTO

To install Objectivity / FTO:

1. Log in to your workstation as `root`, if required for mounting a CD.
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” above).
3. Place the Objectivity distribution CD in the CD-ROM drive and make sure the CD 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` architectures, use the `-o cdcase` option.

4. Run the installation script provided on the distribution CD. Enter:
   `cdMountDir``/install.csh`
   where *`cdMountDir`* is the CD 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 every product you install.

---

6. At the directory prompt, specify the directory in which Objectivity / DB is installed (*`installDir`*). If you get an error message reporting insufficient disk space or incorrect permissions, press Enter to exit the installation script; then fix the problem and restart the installation script (see step 4).
   The installation script:
   - Places the release files in subdirectories of *`installDir/arch`*, where *`arch`* is the architecture name for your platform (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 23.
7. If you also have Objectivity / Smalltalk for VisualWorks installed, follow the steps in "Setting Up Objectivity / Smalltalk for VisualWorks" on page 61.
8. Read:
   - The *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the PDF file *`installDir`*`/doc/ReleaseNotes.pdf`.
   - The 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 *Objectivity/FTO and Objectivity/DRO* online book.
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 Applications to Objectivity/DB” on page 71).

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.

9

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 18).
- Objectivity / FTO (see Chapter 8)

Installing Objectivity/DRO

To install Objectivity / DRO:

1. Log in to your workstation as `root`, if required for mounting a CD.
2. Verify that all required software has been completely and correctly installed (see "Software Requirements" on page 63).
3. Place the Objectivity distribution CD in the CD-ROM drive and make sure the CD 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` architectures, use the `-o cdcase` option.
4. Run the installation script provided on the distribution CD. Enter:

   cdMountDir`/install.csh`

   where *cdMountDir* is the CD 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 every product you install.

---

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

   The installation script:
   - Places the release files in subdirectories of *installDir/arch*, where *arch* is the architecture name for your platform (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 23.
7. If you also have Objectivity / Smalltalk for VisualWorks installed, follow the steps in "Setting Up Objectivity / Smalltalk for VisualWorks" on page 65.
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 18). 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 *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the PDF file *installDir*/doc/ReleaseNotes.pdf.
   - The 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 *Objectivity/FTO and Objectivity/DRO* online book.
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 Applications to Objectivity/DB” on page 71)

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**.
   b. Import the following configuration map into your ENVY server repository from *installDir/arch*/etc/smalltlk/objyDRO.dat.
   Objectivity/DRO
   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/DRO**.
4. Save your image.

10

Objectivity/IPLS Installation

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

System Requirements

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

Software Requirements

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

- Objectivity / DB (see Chapter 1)

Installing Objectivity/IPLS

To install Objectivity / IPLS:

1. Log in to your workstation as `root`, if required for mounting a CD.
2. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 67).
3. Place the Objectivity distribution CD in the CD-ROM drive and make sure the CD 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` architectures, use the `-o cdcase` option.

4. Run the installation script provided on the distribution CD. Enter:
   cdMountDir/install.csh
   where *cdMountDir* is the CD mount directory—for example, /cdrom.
5. At the product prompt, specify Objectivity / IPLS by typing its item number from the displayed list.

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

---

6. At the directory prompt, specify the directory 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 23.
7. Read:
   - The *Objectivity Release Notes* for new and changed features. Use Acrobat Reader to display the PDF file *installDir*/doc/ReleaseNotes.pdf.
   - The 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/IPLS

When you install Objectivity / IPLS, its file is placed in a subdirectory of the *installDir*/*arch* directory, as shown in Table 10-1.

Table 10-1: Objectivity/IPLS Release Files in *installDir/arch*

Subdirectory	File[a]	Description
lib	liboo_ipls.*x.x*.so	Objectivity/IPLS shared library

a. In library names, the digits *x.x* correspond to the current Objectivity/DB release. These digits are omitted on the linux86 architecture.

Using Objectivity/IPLS

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

An application that starts an in-process lock server must be linked with the shared Objectivity / DB library, not the static library (see "Linking Applications to Objectivity / DB" on page 71). No extra steps are required to compile and link an IPLS application. The Objectivity / IPLS shared library listed in Table 10-1 is loaded automatically at runtime when the application starts the in-process lock server.

When an in-process lock server is started, the application that starts it becomes the lock server for the workstation on which it is running, and you must stop any other lock server process that is running on the same workstation. For information about managing in-process and standard lock servers, see Chapter 7, "Using a Lock Server," in the Objectivity / DB administration book.

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 with Objectivity/DB and other Objectivity products
- Using makefiles
- Application programming issues
- Debugging applications

Linking Applications to Objectivity/DB

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

Libraries for Static Linking

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

Table A-1: Objectivity/DB Static Link Libraries

Library File	Description
`liboo.a`	Objectivity/DB standard library.
`liboo_adm.a`	Objectivity/DB administration library. Add this library to your link line *before* `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 *instead of* `liboo.a` if you want to use debug mode or the Objectivity/DB debugger tools (see page 78). 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.x* corresponding to the current Objectivity / DB release. To link Objectivity / DB applications to shared libraries on UNIX, choose a link rule depending on:

- Whether you are linking applications for development or end-user deployment.
- The architecture you are using—for example, `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 75).

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.x	Digits corresponding to the current Objectivity/DB release

Table A-2: Shared Library Link Rules for Development

alphaosf1	-Wl,-call_shared -L*installDir*/alphaosf1/lib -loo.*x.x* -lpthread
hprisc	-L*installDir*/hprisc/lib -loo.*x.x* -lpthread
ibmrs6000	-L*installDir*/ibmrs6000/lib -loo.*x.x* -brtl
iris	-n32 -mips3 -L*installDir*/iris/lib -loo.*x.x* -lpthread
linux86	-L*installDir*/linux86/lib -loo -lrpcsvc -lnsl -ldl -lpthread
solaris4	-L*installDir*/solaris4/lib -loo.*x.x* -mt
solaris7	-L*installDir*/solaris7/lib -loo.*x.x* -mt

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:

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

Table A-3: Shared Library Link Rules for Deployment

alphaosf1	-L*installDir*/alphaosf1/lib -loo.*x.x* -lpthread Use the -rpath linker option to specify *endUserInstallDir*.
hprisc	-L*installDir*/hprisc/lib -loo.*x.x* -Wl,+s -Wl,+b*endUserInstallDir* -lpthread See the ld(1) man page for more information.
ibmrs6000	-L*installDir*/ibmrs6000/lib -L*endUserInstallDir* -loo.*x.x* -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*installDir*/iris/lib -loo.*x.x* -lpthread Use the -rpath linker option to specify *endUserInstallDir*.
linux86	-L*installDir*/linux86/lib -loo -lrpcsvc -lnsl -ldl -lpthread
solaris4	-L*installDir*/solaris4/lib -loo.*x.x* -mt Define LD_RUN_PATH prior to compiling and linking.
solaris7	-L*installDir*/solaris7/lib -loo.*x.x* -mt 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 (`xlC_r`) instead of the normal version (`xlC`) when linking your application. For example:

```
xlC_r -o program a.o b.o c.o
   -LinstallDir/ibmrs6000/lib -loo.x.x -brtl
```

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

ld: 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/xlC/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/xlC/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 CD.

Linking to Additional Objectivity Products and Features

If your database 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 may contain digits *x.x* corresponding to the current Objectivity / DB release.

Table A-4: Objectivity Object Modules and Libraries

To Use	Link to
Objectivity/DRO *and* Objectivity/FTO	`ooRepl.o`
Objectivity/FTO *only*	`ooPart.o`
Objectivity/IPLS	Link with the Objectivity/DB shared library (not `liboo.a`). The Objectivity/IPLS library is loaded automatically at runtime when needed.
Objectivity/C++ persistent collections	`-loo_co` On the `linux86` architecture `-loo_co.x.x` On all other architectures Link with the Objectivity/DB shared library (not `liboo.a`).

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

To Use	Link to	

Objectivity/C++ Active Schema	`-loo_as -lospace`
	`-loo_as.`*`x.x`*
	`-loo_as.`*`x.x`* `-lospace`
		On the `linux86` architecture
		On `iris` and `solaris7`
		On all other architectures

	Link with the Objectivity/DB shared library (not `liboo.a`).	
Objectivity/C++ STL	`libobjystl.a` `libospace.a`	Objectivity/C++ STL library ObjectSpace STL library
	Link to *both* libraries. To do this, include the `local.cfg` file in your makefile, then add `libobjystl.a` and the `OLIBS` variable to your link line (see “Using Makefiles” on page 77).	
Objectivity/SQL++	`-looakit.`*`x.x`* `libooakit.a`	On the `iris` architecture On all other architectures

Linking a Lock-Server Performance-Monitoring Program

Table A-5 lists the libraries for linking a custom C++ program that monitors how your database applications interact with a running lock server. The information collected by such a program can help you analyze application performance (for more information, see the *Monitoring Lock-Server Performance* online book).

You can link the program to either the static or shared library in Table A-5. If you link to the static library, you must also link to `liboo.a`. Otherwise, if you link to the shared library, you must also link to the Objectivity / DB shared library using the appropriate link rule given in Table A-2.

Table A-5: Linking a Lock-Server Performance-Monitoring Program

Link to	Description
`liboolspm.a`	Static runtime library
`-loolspm` `-loolspm.`*`x.x`*	Shared library on the `linux86` architecture Shared library on all other architectures

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. The `csh` and `ksh` commands for increasing the limit are shown in the following example. 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++ programmer's guide).

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

Preparing to Debug an Application

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

Printing Handles

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

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

```
print oopvar
```

The variable `oopvar` is printed as follows:

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

Printing Objects

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

B

Java Application Development

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

`-Xss`*`size`*

where

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

EXAMPLE To specify a stack size of 2 megabytes:

```
% java -Xss2m classname
```

File Descriptor Limit

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

To eliminate these errors, you should increase the file descriptor limit. The `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
```

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:

`-Xms`*`size`*	Sets the initial memory size.
`-Xmx`*`size`*	Sets the maximum memory size.

where

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

C

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

Solution:

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

Object Does Not Open

Solutions:

- Verify that a lock server is running on the node specified by the `ooLockServerName` value in the boot file.
- Check whether a network failure is preventing access to the node where the lock server is running.
- If a `dbx` session was terminated while debugging an application, check if any locks remain**.**
- If your application 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

F

H

I

L

M

N

O

P

R

S

T

U

V

X