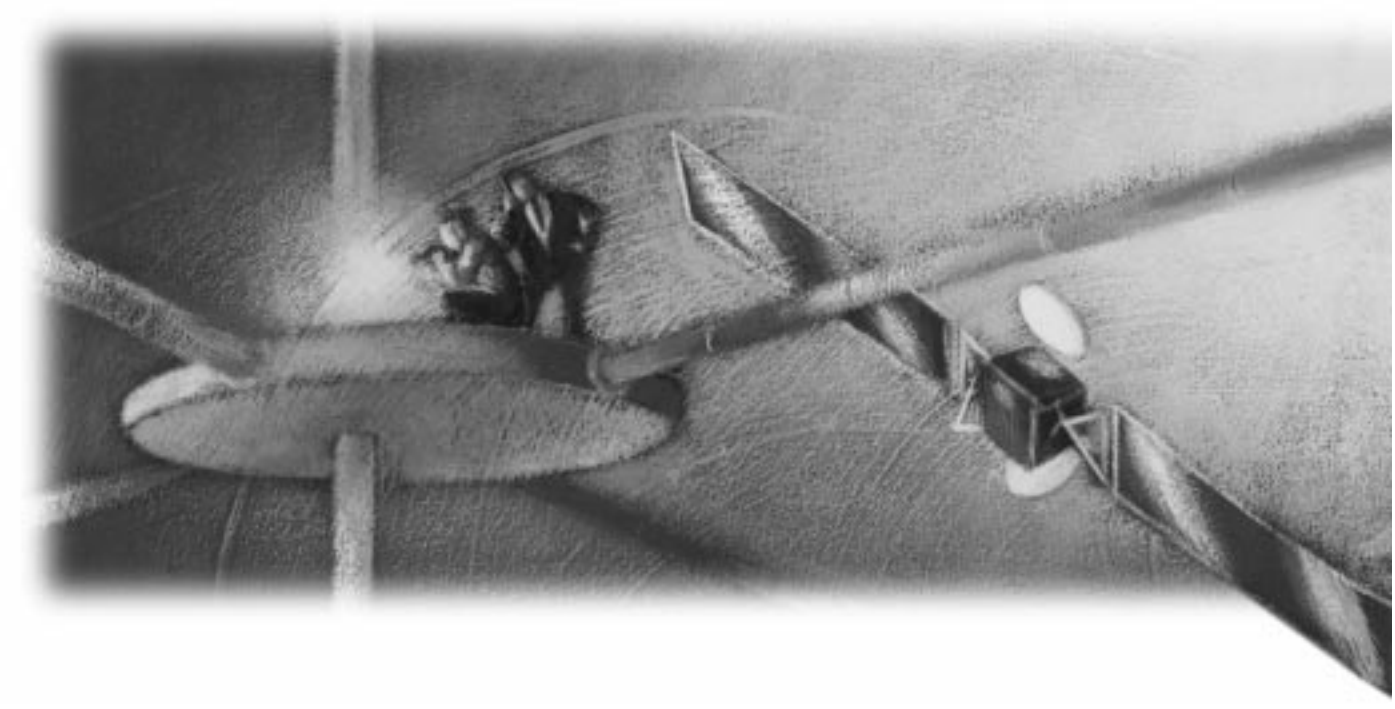


Installation and Platform Notes for Windows

Release 6.0

Installation and Platform Notes for Windows

Part Number: 60-IWIN-0

Release 6.0, October 20, 2000

Contents

About This Book

This book, *Installation and Platform Notes for Windows*, describes how to install Objectivity products on computers running supported Microsoft Windows operating systems. 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 familiarity 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 Windows. The appendixes assume familiarity with the compiler and development environment.

Organization

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

Appendix A provides platform-specific details that supplement the information in the books for Objectivity / C++ and its options. Topics include linking, creating DLLs that use Objectivity / DB, C++ programming issues, and building and debugging C++ database applications.

Appendix B describes how to uninstall Objectivity products.

Appendix C provides suggestions for troubleshooting an Objectivity / DB application.

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 Windows 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 Windows platforms listed in Table 1-1.

See the release notes on the Objectivity Technical Support web site for the currently supported versions of the Windows operating system. Contact Objectivity Customer Support to get access to this web site.

Table 1-1: Supported Windows Platforms for Objectivity/DB

Hardware	Operating System	Abbreviation[a]
Intel Pentium or greater	Windows 98	Windows 98
	Windows NT Workstation Windows NT Server	Windows NT
	Windows 2000 Professional Windows 2000 Server	Windows 2000

a. These abbreviations are used in Objectivity books to identify the corresponding platforms.

Software Requirements

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

- Winsock-compatible TCP / IP software (see “Verifying and Configuring TCP / IP” on page 21)

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`.

On Windows NT and Windows 2000, you must have a logon account (such as administrator) that has privileges to start, stop, and configure network services.

Hardware Requirements

Table 1-2 shows the recommended hardware configuration for installing Objectivity / DB, Objectivity / FTO, Objectivity / DRO, and one of the Objectivity programming interfaces (Objectivity / C++, Objectivity for Java, or Objectivity / Smalltalk for VisualWorks) on a Windows platform. The setup program reports the amount of free disk space required for the specific products you are installing, and determines whether your computer has enough.

Table 1-2: Recommended Hardware Configuration

Hardware	Recommended
Processor	166-MHz Pentium or higher
RAM	40 MB (Objectivity/C++ or Objectivity for Java) 64 MB (Objectivity/Smalltalk for VisualWorks)
Free disk space	100 MB

Installing Objectivity/DB

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

NOTE On Windows NT or Windows 2000, you must log on as administrator or as a user with equivalent privileges.

---

1. Verify that TCP / IP is installed, running, and configured properly (see "Verifying and Configuring TCP / IP" on page 21).
2. If you are installing multiple Objectivity products at this time, verify that the system requirements for those products are met (see the installation chapter for each product in this book).
3. If you have any Release 5.*x* Objectivity products currently installed on your computer, consider uninstalling them (see Appendix B, "Uninstalling Objectivity Products"). At a minimum, you *must* ensure that no Release 5.2 Objectivity servers are being used by active applications because these servers will be replaced by Release 6.0 servers during installation (see "Maintaining Older Objectivity / DB Releases" on page 20).
4. Place the Objectivity distribution CD in your computer's CD-ROM drive. The setup program for installing Objectivity products will start automatically.

NOTE If you need to start the setup program explicitly, display the CD-ROM drive and double-click `setup.exe` on the CD.

---

5. Select the directory (*`installDir`*) in which to install Objectivity / DB and any other Objectivity products. The default is `c:\objy60`.
6. Select **Objectivity/DB** and any other Objectivity products you want to install at this time. If you select a product that requires other products, the required products are selected automatically.

NOTE You must have a license for each Objectivity product to be installed.

---

7. Click **Next** and follow the prompts to complete the installation.
8. When prompted, restart your computer so that the setup program can update the system registry.
9. Verify that Objectivity servers have been successfully installed (see "Verifying Objectivity Server Status" on page 16).

10. On every Windows 98 computer that is to run an Objectivity / DB application or store an Objectivity / DB file, make sure that write caching is disabled. In Control Panel, select:

    System > Performance > File System > Troubleshooting > Disable write-behind caching

11. 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 18. You may need to upgrade the indexes or schemas of such federated databases before you can access them.
12. If you plan to continue using tools or applications built with earlier Objectivity / DB releases, read "Maintaining Older Objectivity / DB Releases" on page 20. You may need to change the host on which you run an older version of the lock server.
13. If you are installing multiple Objectivity products at this time, read the installation chapters for those products and follow any additional steps for setting them up.
14. Familiarize yourself with Objectivity online books. To do so, click **Start** and point to **Programs**; in the Objectivity submenu, click **Objectivity Books**. A PDF file is displayed containing links to the online books.

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

---

15. Read:
    - *Objectivity Release Notes* for new and changed features. (Click **Start** and point to **Programs**. In the Objectivity submenu, click **About This Release**.)
    - 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.

What Objectivity/DB Setup Does

For Objectivity / DB, the setup program:

- Creates an Objectivity submenu under **Programs** in the **Start** menu, and adds the program shortcuts shown in Table 1-3.

Table 1-3: Program Shortcuts Installed by Objectivity Setup

Shortcut	Description
About This Release	Displays *Objectivity Release Notes* (in PDF), which describes new and changed Objectivity products and documentation
Objectivity Books	Displays a starting point for viewing the online books (in PDF) for Objectivity products
Objectivity Network Services	Starts a tool for managing Objectivity server processes
ObjyTool	Starts a tool for running database administration tools and processes
oobrowse	Starts a tool for browsing objects and types in an Objectivity/DB federated database

- Installs files in subdirectories of the Objectivity / DB installation directory *`installDir`*, as shown in Table 1-4.

Table 1-4: Objectivity/DB Release Files in *`installDir`*

Subdirectory	Contains
`bin`	Executables for Objectivity/DB tools
	Executables for the Objectivity/DB servers
	Dynamic link library (DLL) for Objectivity/DB tools and the Objectivity/DB kernel
	DLL for custom lock-server performance-monitoring programs
`doc`	PDF files for the online books of Objectivity products PDF file displayed by the **Objectivity Books** shortcut (contains links to the online books)
`etc`	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`	Link library for custom lock-server performance-monitoring programs

- Creates entries for Objectivity servers in the TCP/IP `services` file, and then starts these servers (see "Verifying Objectivity Server Status" on page 16).
- Modifies the system registry to set environment variables shown in Table 1-5.

Table 1-5: Basic Environment Variables Modified by Setup Program

Variable	Value
`include`	Search path for include files; used by Objectivity/C++ and Microsoft Visual C++. Set to contain *`installDir`*`\include`, which is created by Objectivity/C++. This component must precede the Visual C++ include directory in the search path.
`lib`	Search path for library files; used by Objectivity/C++ and Microsoft Visual C++. Modified to contain *`installDir`*`\lib`, which is created by Objectivity/C++. This component must precede the Visual C++ library directory in the search path.
`path`	Search path for executables, including Objectivity/DB administration and database tools. Modified to contain *`installDir`*`\bin`.

Verifying Objectivity Server Status

The setup program installs and starts the two Objectivity servers:

- The *lock server*, which manages concurrent access to persistent objects in one or more federated databases
- The Advanced Multithreaded Server (AMS), which provides access to remote database files in a distributed Objectivity/DB system

You manage these servers using the Objectivity Network Services tool. These servers are set up to start automatically whenever the computer boots and to run even when no user is logged in.

To verify that the Objectivity servers were installed and set up successfully:

1. On Windows NT or Windows 2000, log on as administrator or as a user with equivalent privileges.
2. Click **Start** and point to **Programs**. In the Objectivity submenu, select **Objectivity Network Services**.
3. Verify that each server is listed as installed and running.

If an Objectivity Server is Stopped

If, after installation, the Objectivity Network Services tool reports that one or both Objectivity servers are stopped, you should check whether another process is already running on the port that the server expected to use.

Each Objectivity server is assigned a default TCP/IP port number by Objectivity/DB. If another service already uses one of the default ports, you should try to reassign that service to a different port. If you cannot do this, you may have to change the default port for the Objectivity server. To confirm that a port conflict exists or to change the default port for an Objectivity server, see Chapter 7, "Using a Lock Server," or Chapter 8, "Advanced Multithreaded Server," in the Objectivity/DB administration book.

NOTE If you change the port number for an Objectivity server, you must make this change on every host that is to run a process that uses the server.

If the Lock Server is Running

If the lock server is running, you can let it run or you can stop it using the Objectivity Network Services tool. The lock server should normally remain running. If you stop the lock server, you must restart it before running any database application that requires concurrency management.

On Windows NT or Windows 2000, the lock server is started under the local system account by default. For security purposes, you should consider running the lock server under a special-purpose user or group account that can be granted the necessary permissions and no others. Furthermore, if you plan to access remote database files using Universal Naming Convention (UNC) names, the lock server must run under an account that has permissions for the appropriate shared drives. For information about starting a lock server with the required permissions, see Chapter 7, "Using a Lock Server," in the Objectivity/DB administration book.

If AMS is Running

If AMS is running, you can let it run or you can stop it using the Objectivity Network Services tool. AMS should remain running if you choose to use it for remote data access.

In general, AMS is recommended for its performance, flexibility, and ease of use. In some cases, different software, such as the Network File System (NFS) or Microsoft Windows Network, may be preferable. To help you choose the data server software that is appropriate for your Windows hosts, see Chapter 8, "Advanced Multithreaded Server," in the Objectivity/DB administration book.

On Windows NT or Windows 2000, AMS is started under the local system account by default. For security purposes, you should consider running AMS under a special-purpose user or group account that can be granted the just the minimum necessary permissions. When an application uses AMS, any database files created by the application are owned by the user account under which AMS was started. For information about starting AMS with the required permissions, see Chapter 8, “Advanced Multithreaded Server,” in the Objectivity / DB administration book.

If You Choose NFS Instead of AMS

If you choose to use NFS instead of AMS for remote data access, 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` in the registry.

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	19
Schema	Any release prior to Release 5.2	19
(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 5.2, you must upgrade its schema. You must perform this upgrade before you create or access persistent collections from an application written in Objectivity / C++, Objectivity for Java, or Objectivity / Smalltalk for VisualWorks.

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\etc\ooCollectionsSchema.dmp
    bootFilePath
```

where

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

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 (for example, by installing Objectivity / DB Release 6.0 on that 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 computers. 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.

Verifying and Configuring TCP/IP

Objectivity / DB relies on the availability of a TCP / IP communications protocol that conforms to the Winsock specification. Basic TCP / IP services are required by all Objectivity / DB configurations, even completely local configurations (where all Objectivity / DB files, servers, and applications are on the same computer).

Windows 98

Microsoft TCP / IP is included with Windows 98, so you do not need to purchase additional software.

Your computer must have an installed network adapter card or a modem, even if you do not connect to a network.

To verify and configure Microsoft TCP / IP on Windows 98:

1. Verify that TCP / IP is installed on your computer. To do this, you can click **Start**, point to **Programs**, click **MS DOS Prompt**, and run the `winipcfg` command.

   If TCP / IP is installed, a dialog displays the current settings; otherwise, an error message appears.
2. If necessary, install TCP / IP according to the Windows 98 online help (look up `installing network protocols`). Ask your system administrator for the IP address and any other necessary settings for your computer.
3. After TCP / IP is installed, edit the TCP / IP hosts configuration file (by default, `C:\windows\hosts`) and make sure entries exist for your computer and any other computers you wish to access through Objectivity / DB. Entries should include the hostname and IP address. You can dramatically boost performance by putting entries in your `hosts` file, even if you are using DNS.
4. If you are using DHCP, ask your system administrator to set up a DHCP reservation for your host to ensure that the same IP address will always be assigned to your host.
5. Restart your computer.

Windows NT and Windows 2000

Microsoft TCP/IP is included with Windows NT and Windows 2000, so you do not need to purchase additional software.

Your computer should have an installed network adapter card or loopback adapter, even if you do not connect to a network. To obtain a loopback adapter:

- (Windows NT) In Control Panel, click **Network Settings > Add Adapter** and choose the Microsoft loopback adapter.
- (Windows 2000) Install the Microsoft loopback adapter using the steps in article Q236869 in the Microsoft Searchable Knowledge Base.

To configure Microsoft TCP/IP on Windows NT and Windows 2000:

1. Log on as administrator or as a user with equivalent privileges.
2. Verify that TCP/IP is installed on your computer. To do this, you can run the `ipconfg` command in a command prompt. To get a command prompt:
   - (Windows NT) Click **Start**, point to **Programs**, click **Command Prompt**.
   - (Windows 2000) Click **Start**, point to **Programs**, point to **Accessories**, and click **Command Prompt**.

   If TCP/IP is installed, a dialog displays the current settings; otherwise, an error message appears.
3. If necessary, install TCP/IP according to the online help for Windows NT or Windows 2000 (look up `installing TCP/IP`). Ask your system administrator for the IP address and any other necessary settings for your computer.
4. After TCP/IP is installed, edit the TCP/IP hosts configuration file (by default, `C:\winnt\system32\drivers\etc\hosts`) and make sure entries exist for your computer and any other computers you wish to access through Objectivity/DB. Entries should include the hostname and IP address. You can dramatically boost performance by putting entries in your `hosts` file, even if you are using DNS.
5. If you are using DHCP, ask your system administrator to set up a DHCP reservation for your host to ensure that the same IP address will always be assigned to your host.

2

Objectivity/C++ Installation

This chapter describes the requirements and steps for installing Objectivity / C++ on a Windows 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 schemas of persistent C++ data types in Objectivity / DB databases. The DDL processor also produces source and header files for these data types.

System Requirements

You can install Objectivity / C++ on the Windows platforms listed in Table 1-1 on page 11.

Software Requirements

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

- Objectivity / DB (see Chapter 1)
- Microsoft Visual C++

  See the release notes on the Objectivity Technical Support web site for the currently supported C++ compiler version. Contact Objectivity Customer Support to get access to this web site.

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

Hardware Requirements

Table 1-2 on page 12 lists the recommended hardware requirements for installing Objectivity / C++ and Objectivity / DDL along with Objectivity / DB.

Installing Objectivity/C++ or Objectivity/DDL

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

1. Verify that all required software has been completely and correctly installed (see "Software Requirements" on page 23).
2. Place the Objectivity CD in your CD-ROM drive. The setup program for installing Objectivity products will start automatically.

   If you need to start the setup program explicitly, display the CD-ROM drive and double-click `setup.exe` on the CD.
3. Click **Modify** and then click **Next** to display a list of Objectivity products; a check mark ✓ next to a product indicates that the product is already installed.
4. Select **Objectivity/C++**. To also install Objectivity / DDL, select **Objectivity/C++ Data Definition Language**. If a required product is not yet installed, it is selected and installed automatically.

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

---

5. Click **Next** and follow the prompts to complete the installation.
6. (Optional) Test the installation of Objectivity / C++ and Objectivity / DDL by running the provided C++ demo applications (see "Testing Objectivity / C++ Setup" on page 25).
7. (Optional) Set up the Visual C++ Integrated Development Environment (IDE) for use with Objectivity / C++ (see "Setting Up the Visual C++ IDE" on page 27).
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 19).
9. Read:
   - Appendix A, "C++ Application Development," in this book for platform-specific information about using Objectivity / C++.
   - *Objectivity Release Notes* for new and changed features. (Click **Start** and point to **Programs**. In the Objectivity submenu, click **About This Release**.)

- 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.

What Setup Does

For Objectivity / C++ and Objectivity / DDL, the setup program:

- Installs files in subdirectories of the Objectivity / DB installation directory *`installDir`*, as shown in Table 2-1.

Table 2-1: Objectivity/C++ Release Files in *`installDir`*

Subdirectory	Contains
`bin`	Executable for the DDL processor (Objectivity/DDL)
	Dynamic link libraries for C++ database applications (see “Linking Applications to Objectivity/DB” on page 73)
`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`	Link libraries for user-created C++ database applications (see “Linking Applications to Objectivity/DB” on page 73)
`samples`	C++ database applications for demonstration and testing (see “Testing Objectivity/C++ Setup” on page 25)

- Verifies the settings of the `path`, `lib`, and `include` environment variables. If necessary, these variables should be set to the values shown in Table 1-5.
- Adds the Objectivity / DB `bin`, `include`, and `lib` directories to the Visual C++ IDE search path.

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 application provided with the installation. You can build this demo application either using the Visual C++ IDE, or from a command prompt. The demo application generates a federated database and interacts with it using the C++ interface. You can also inspect the demo application to see how to use various Objectivity / C++ features.

The demo subdirectory *installDir*\samples\cppdll contains:

- A Visual C++ project file for use with the Visual C++ IDE
- A standard makefile (called makefile) for use with nmake

Building the Demo Application Using the Visual C++ IDE

To build and run a demo application using the Visual C++ IDE:

1. Set up the Visual C++ IDE to work with Objectivity / DB (see "Setting Up the Visual C++ IDE" on page 27).
2. Select **File > Open Workspace** and specify the project in the *installDir*\samples\cppdll directory.
3. Select **Build > Rebuild All**.
4. See "Demo Results" on page 26.

Building the Demo Application From a Command Prompt

To build and run a demo application from a command prompt:

1. Check your environment variable settings (see Table 1-5 on page 16).
2. Change to the *installDir*\samples\cppdll directory.
3. Build the application. Enter:
   ```
   nmake
   ```
4. See "Demo Results" on page 26.

Demo Results

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

```
Hello, world!
Welcome to Objectivity/DB!
The installation test has PASSED.
```

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

```
The installation test has FAILED.
```

If the installation test failed, verify the makefile, the compiler, and the settings of the environment variables. Correct any errors and build the demo application again. If another user has already run the demo program, enter `nmake cleandb` to reset the demo directory before building the demo application. If the application still fails, contact Objectivity Customer Support for assistance.

Setting Up the Visual C++ IDE

You can set up the Visual C++ IDE to work with Objectivity / C++ and Objectivity / DDL. The following subsections describe basic setup steps for the Visual C++ IDE. After you have performed the basic setup for the Visual C++ IDE, you can:

- Run the Objectivity / C++ demo applications (see "Building the Demo Application Using the Visual C++ IDE" on page 26).
- Set up your own projects by following the steps in "Visual C++ IDE-Based Development" on page 85.

Setting the Visual C++ IDE Search Path

The setup program automatically adds the Objectivity / DB `bin`, `include`, and `lib` directories to the Visual C++ IDE search path. If, however, you ran the setup program under a different logon account than you normally use, you may need to set the search path under your normal logon account. To do this:

1. On Windows NT or Windows 2000, log on using your logon account.
2. Select **Tools > Options > Directories**.
3. In the Show Directories For list, select **Executable files** and enter the following path in the Directories list:
   `installDir``\bin`
4. In the Show Directories For list, select **Include files** and enter the following path in the Directories list:
   `installDir``\include`
5. In the Show Directories For list, select **Library files** and enter the following path in the Directories list:
   `installDir``\lib`

Defining Actions on the Visual C++ IDE Tools Menu

You can optionally define various Objectivity / C++ and Objectivity / DDL actions on the Visual C++ IDE **Tools** menu. The steps in this section show you how to define the actions listed in Table 2-2. (You can add other Objectivity / C++ actions to the **Tools** menu in a similar manner.)

Table 2-2: Objectivity/C++ Actions to be Added

Item on Tools Menu	Action
Build Schema	Runs the DDL processor only if the schema DDL file has been modified.
Rebuild Schema	Always runs the DDL processor.
List FD Files	Runs the Objectivity/DB `oodumpcatalog.exe` tool.

For each action to be defined in the Visual C++ IDE **Tools** menu, you:

1. Select **Tools > Customize > Tools**.
2. Click the icon for adding a new action. In the corresponding input fields, enter the values for the desired action shown in Table 2-3.

Table 2-3: Values for Defining Objectivity/C++ Actions

Menu Contents	Command	Arguments	Initial Directory
`Build Schema`	`nmake`	`-f makefile.mvc ddl_build`	`$(WkspDir)`
`Rebuild Schema`	`nmake`	`-f makefile.mvc /a ddl_build`	`$(WkspDir)`
`List FD Files`	`nmake`	`-f makefile.mvc dumpcat`	`$(WkspDir)`

3. Click the Use Output Window check box and click **OK**.
4. Ensure that each project directory where you intend to run these tools contains an Objectivity / C++-specific makefile called `makefile.mvc`. Note that:
   - Each demo subdirectory in *`installDir`*`\samples` already contains an Objectivity / C++-specific makefile.
   - When setting up your own project directory, you can copy the `makefile.mvc` and `makefile` files from any demo subdirectory—for example, *`installDir`*`\samples\cppdll`. You can then customize the `makefile` file as appropriate to your project (see “Creating a Custom Makefile for Your Project” on page 85).

3

Objectivity/C++ STL Installation

This chapter describes the requirements and steps for installing Objectivity / C++ Standard Template Library (Objectivity / C++ STL) on a Windows 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 Windows platforms listed in Table 1-1 on page 11.

Software Requirements

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

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

Installing Objectivity/C++ STL

To install Objectivity / C++ STL:

1. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 29).
2. Place the Objectivity CD in your CD-ROM drive. The setup program for installing Objectivity products will start automatically.

   If you need to start the setup program explicitly, display the CD-ROM drive and double-click `setup.exe` on the CD.

3. Click **Modify** and then click **Next** to display a list of Objectivity products; a check mark ✓ next to a product indicates that the product is already installed.
4. Select **Objectivity/C++ Standard Template Library**. If a required product is not yet installed, it is selected and installed automatically.

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

5. Click **Next** and follow the prompts to complete the installation.
6. Read:
   - *Objectivity Release Notes* for new and changed features. (Click **Start** and point to **Programs**. In the Objectivity submenu, click **About This Release**.)
   - 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.

What Objectivity/C++ STL Setup Does

For Objectivity / C++ STL, the setup program:

- Creates the `ToolKit` subdirectory in the Objectivity / DB installation directory *`installDir`* and installs files in subdirectories of `ToolKit`, as shown in Table 3-1.

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

Subdirectory of `ToolKit`	Contains
`config`	Configuration information
`doc`	ObjectSpace STL online documentation
`lib`	ObjectSpace STL and Objectivity/C++ STL import libraries and DLLs
`msvc6.0`	ObjectSpace project files
`ospace\stl`	ObjectSpace STL and Objectivity/C++ STL include files and source files
`ospace\stl\d_examples`	Objectivity/C++ STL example applications

- Adds the Objectivity / C++ STL *`installDir`*`\ToolKit` and *`installDir`*`\ToolKit\lib` directories to the Visual C++ IDE search path.
- Adds the PDF file for the *Objectivity/C++ Standard Template Library* online book in the *`installDir`*`\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.

You build the demo applications through the Visual C++ IDE, using the provided project file and makefile:

1. Verify that the Visual C++ IDE is set up to work with Objectivity / C++ and Objectivity / DDL (see "Setting Up the Visual C++ IDE" on page 27).
2. If necessary, add the Objectivity / C++ STL directories to the Visual C++ IDE search path (you may need to do this if you installed Objectivity / C++ STL under a different logon account):
   a. Select **Tools > Options > Directories**.
   b. In the Show Directories For list, select **Include files** and enter the following path in the Directories list:
   installDir\ToolKit
   c. In the Show Directories For list, select **Library files** and enter the following path in the Directories list:
   installDir\ToolKit\lib
3. Select **File > Open Workspace** and specify the `objyexamples.dsw` project in the *installDir*`\ToolKit\ospace\stl\d_examples` directory. This project has a subproject for each demo application.
4. Select **Tools > Rebuild Schema** to create a federated database and invoke the DDL processor. Alternatively, if you have not added this action to your **Tools** menu, you can enter the following in a command prompt:
   ```
   nmake -f makefile.mvc /a ddl_build
   ```
5. Select **Build > Batch Build > Rebuild All** to build the entire set of demo applications, and verify that no error messages appear in the Visual C++ IDE output window.

At this point, you can run each demo application individually.

4

Objectivity/C++ Active Schema Installation

This chapter describes the requirements and steps for installing Objectivity / C++ Active Schema (Objectivity / AS) on a Windows 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 the Windows platforms listed in Table 1-1 on page 11.

Software Requirements

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

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

Installing Objectivity/AS

To install Objectivity / AS:

1. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 33).
2. Place the Objectivity CD in your CD-ROM drive. The setup program for installing Objectivity products will start automatically.

   If you need to start the setup program explicitly, display the CD-ROM drive and double-click `setup.exe` on the CD.
3. Click **Modify** and then click **Next** to display a list of Objectivity products; a check mark ✓ next to a product indicates that the product is already installed.

4. Select **Objectivity/C++ Active Schema**. If a required product is not yet installed, it is selected and installed automatically.

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

5. Click **Next** and follow the prompts to complete the installation.
6. Read:
   - *Objectivity Release Notes* for new and changed features. (Click **Start** and point to **Programs**. In the Objectivity submenu, click **About This Release**.)
   - 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.

What Objectivity/AS Setup Does

For Objectivity / AS, the setup program installs files in subdirectories of the Objectivity / DB installation directory *`installDir`*, as shown in Table 4-1.

Table 4-1: Objectivity/AS Release Files in *`installDir`*

Subdirectory	Contains
`bin`	Dynamic link libraries for Objectivity/AS (see “Linking to Objectivity/C++ Active Schema” on page 80)
`doc`	PDF file for *Objectivity/C++ Active Schema* online book
`include`	Include file `ooas.h` for the Objectivity/AS programming interface
`lib`	Link libraries for Objectivity/AS (see “Linking to Objectivity/C++ Active Schema” on page 80)

5

Objectivity for Java Installation

This chapter describes the requirements and steps for installing Objectivity for Java on a Windows 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 Windows platforms listed in Table 1-1 on page 11.

Software Requirements

Objectivity for Java requires that the following software be installed on your computer:

- Objectivity / DB (see Chapter 1)
- Java 2 Software Development Kit (SDK)

  Note: See the release notes on the Objectivity Technical Support web site for the currently supported SDK version. 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.

Hardware Requirements

Table 1-2 on page 12 lists the recommended hardware requirements for installing Objectivity for Java along with Objectivity / DB.

Installing Objectivity for Java

To install Objectivity for Java:

1. Verify that all required software has been completely and correctly installed (see "Software Requirements" on page 35).
2. Place the Objectivity CD in your CD-ROM drive. The setup program for installing Objectivity products will start automatically.

   If you need to start the setup program explicitly, display the CD-ROM drive and double-click `setup.exe` on the CD.
3. Click **Modify** and then click **Next** to display a list of Objectivity products; a check mark ✓ next to a product indicates that the product is already installed.
4. Select **Objectivity for Java**. If a required product is not yet installed, it is selected and installed automatically.

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

---

5. Click **Next** and follow the prompts to complete the installation.
6. Add the Objectivity for Java library path, *`installDir`*`\lib\oojava.jar`, to the `CLASSPATH` environment variable. You can set `CLASSPATH` at the command line, by using a batch file, or by declaring `CLASSPATH` as a variable within the environment settings.

   For some application development environments (such as Visual J++ or Visual Café), you must specify `CLASSPATH` from within the tool.
7. 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 37).
8. 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 19).
9. Read:
   - *Objectivity Release Notes* for new and changed features. (Click **Start** and point to **Programs**. In the Objectivity submenu, click **About This Release**.)
   - 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.

What Objectivity for Java Setup Does

The Objectivity for Java setup program:

- Installs files in subdirectories of the Objectivity/DB installation directory *installDir*, as shown in Table 5-1.

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

Directory	Location	Contains
bin	oojava.dll	Library executable
doc	javaGuide.pdf	Online guide (PDF)
	java\index.html	Document index
	java\api*.html	Online reference
	java\guide*.html	Online guide (HTML)
	java\samples\GettingStarted*.java	Sample applications
lib	oojava.jar	Library executable
src	java\src.zip	Library source

- Adds the **Objectivity for Java Books** shortcut to the Objectivity submenu. This shortcut displays an index page that provides access to the HTML online books through your World Wide Web 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 its schema by adding 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 demo 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. The

demo application generates a federated database and interacts with the federated database using the Objectivity for Java interface. You can also inspect the demo 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 Windows 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 Windows platforms listed in Table 6-1.

Table 6-1: Supported Windows Platforms for Objectivity/Smalltalk for VisualWorks

Hardware	Operating System	Abbreviation
Intel Pentium or greater	Windows NT Workstation Windows NT Server	Windows NT
	Windows 2000 Professional Windows 2000 Server	Windows 2000

Software Requirements

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

- 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.

Hardware Requirements

Table 1-2 on page 12 lists the recommended hardware requirements for installing Objectivity / Smalltalk for VisualWorks along with Objectivity / DB.

Installing Objectivity/Smalltalk for VisualWorks

To install Objectivity / Smalltalk for VisualWorks:

1. Verify that all required software has been completely and correctly installed (see "Software Requirements" on page 39).
2. Place the Objectivity CD in your CD-ROM drive. The setup program for installing Objectivity products will start automatically.

   If you need to start the setup program explicitly, display the CD-ROM drive and double-click `setup.exe` on the CD.
3. Click **Modify** and then click **Next** to display a list of Objectivity products; a check mark ✓ next to a product indicates that the product is already installed.
4. Select **Objectivity/Smalltalk for VisualWorks**. If a required product is not yet installed, it is selected and installed automatically.

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

---

5. Click **Next** and follow the prompts to complete the installation.
6. Set up your image by following the instructions in "Setting Up VisualWorks" *or* "Setting Up VisualWorks With ENVY / Developer" on page 42.
7. 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 19).
8. Read:
   - *Objectivity Release Notes* for new and changed features. (Click **Start** and point to **Programs**. In the Objectivity submenu, click **About This Release**.)
   - 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.

What Objectivity/Smalltalk for VisualWorks Setup Does

For Objectivity / Smalltalk for VisualWorks, the setup program installs files in subdirectories of the Objectivity / DB installation directory *installDir*, as shown in Table 6-2.

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

Subdirectory	File	Description
bin	oogc.exe	Garbage collection utility.
	oost*xx*.dll[a]	Objectivity/Smalltalk DLL (bridge to Objectivity/DB kernel).
doc	smalltalk.pdf	PDF file for the *Objectivity/Smalltalk for VisualWorks* online book.
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.

a. The digits *xx* in a DLL name indicate the current Objectivity/DB release.

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*\etc\smalltlk\objyDB.st.
3. Enter the following in the prompt **Please provide a fully qualified filename:**
   installDir\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 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*\etc\smalltlk\objyDB.dat to free disk space. This file is only used with ENVY / Developer.
10. (Optional) Test the installation (see "Testing Objectivity / Smalltalk for VisualWorks Setup" on page 43).

Setting Up VisualWorks With ENVY/Developer

To set up VisualWorks with 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*\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" on page 43).
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 computer 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 Windows 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 Windows platforms listed in Table 7-1.

Table 7-1: Supported Windows Platforms for Objectivity/SQL++

Hardware	Operating System	Abbreviation
Intel Pentium or greater	Windows NT Workstation Windows NT Server	Windows NT
	Windows 2000 Professional Windows 2000 Server	Windows 2000

Software Requirements

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

- 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)
- Objectivity/SQL++ ODBC Driver (see Chapter 8)

Installing Objectivity/SQL++

To install Objectivity/SQL++:

1. Log on as administrator or as a user with equivalent privileges.
2. Verify that all required software has been completely and correctly installed (see "Software Requirements" on page 46).
3. Place the Objectivity CD in your CD-ROM drive. The setup program for installing Objectivity products will start automatically.

   If you need to start the setup program explicitly, display the CD-ROM drive and double-click `setup.exe` on the CD.
4. Click **Modify** and then click **Next** to display a list of Objectivity products; a check mark ✓ next to a product indicates that the product is already installed.
5. Select **Objectivity/SQL++**. If a required product is not yet installed, it is selected and installed automatically.

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

---

6. Click **Next** and follow the prompts to complete the installation.
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 about `systpe`, see the Objectivity/SQL++ book.
8. Set up the Objectivity/SQL++ ODBC server by following the steps in "Setting Up the Objectivity/SQL++ ODBC Server" on page 48.
9. Test each Objectivity/SQL++ component you plan to use:
   - Test Interactive SQL++ by following the steps in "Testing Interactive SQL++" on page 50.

- Test the Objectivity / SQL++ programming interface by following the steps in "Testing the Programming Interface" on page 51.
- 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 52.

10. Read:

- *Objectivity Release Notes* for new and changed features. (Click **Start** and point to **Programs**. In the Objectivity submenu, click **About This Release**.)
- 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.

What Objectivity/SQL++ Setup Does

For Objectivity / SQL++, the setup program:

- Installs files in subdirectories of the Objectivity / DB installation directory *`installDir`*, as shown in Table 7-2.

Table 7-2: Objectivity/SQL++ Release Files in *`installDir`*

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
`samples\sql`	Subdirectories containing applications for demonstration and testing (see "Testing Interactive SQL++" on page 50 and "Testing the Programming Interface" on page 51)

- Installs the Objectivity / SQL++ ODBC server in the **Objectivity Network Services** tool.

Setting Up the Objectivity/SQL++ ODBC Server

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

- Verifying that it is installed and running
- Setting up a boot file directory
- Optionally specifying a nondefault log directory

Verifying the ODBC Server Installation

The setup program installs and starts the Objectivity / SQL++ ODBC server. You manage this server using the Objectivity Network Services tool. The Objectivity / SQL++ ODBC server is set up to start automatically whenever the system boots and to run even when no user is logged in.

To verify that the Objectivity / SQL++ ODBC server was installed successfully:

1. Log on as administrator or as a user with equivalent privileges.
2. Click **Start** and point to **Programs**. In the Objectivity submenu, select **Objectivity Network Services**.
3. Verify that the Objectivity / SQL++ ODBC server is listed as installed and running.

If the ODBC Server is Stopped

If Objectivity Network Services lists the Objectivity / SQL++ ODBC server as stopped, you should perform the following steps:

1. Open the TCP / IP `services` file. By default, this file is `C:\winnt\system32\drivers\etc\services`.
2. Check whether the `services` file contains an entry such as the following:
   ```
   oosqlnw      1990/tcp     # Objectivity/SQL++ ODBC server
   ```
3. If the `oosqlnw` service has an entry, go to step 4; otherwise, edit the `services` file to add the entry shown in step 2. Note that port 1990 is recommended.
4. Check whether another service already uses TCP / IP port 1990—for example by inspecting by opening the Event Viewer from the Administrative Tools folder of the Control Panel, and looking in the Application Log. If a port conflict exists, the message `10048` (`WSAEADDRINUSE`) is displayed.
5. If another service already uses port 1990, either reassign that service to a different port (recommended) or edit the `services` file to change the port for 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.

6. Restart your computer.
7. Verify that the Objectivity/SQL++ ODBC server is running. (Click **Start** and point to **Programs**; in the Objectivity submenu, select **Objectivity Network Services**.)

If the ODBC Server is Running

If Objectivity Network Services lists the Objectivity/SQL++ ODBC server as running, you can let it run or you can stop it by clicking **Stop**. A running ODBC server is required only when you are accessing Objectivity/DB databases with ODBC-compliant applications that use the Objectivity/SQL++ ODBC Driver.

A running ODBC server is *not* required if you are accessing Objectivity/DB databases with Interactive SQL++ or with C++ applications that use the Objectivity/SQL++ programming interface.

Setting Up a Boot File Directory

You must enable the Objectivity/SQL++ ODBC server to locate the federated databases 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 (for example, 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. Specify the boot file directory to the Objectivity/SQL++ ODBC server:
   a. Log on as administrator or as a user with equivalent privileges.
   b. Click **Start** and point to **Programs**. In the Objectivity submenu, select **Objectivity Network Services**.
   c. Select the Objectivity/SQL++ ODBC server and click the **Configure** button.
   d. Enter the pathname of the boot file directory and click **OK**. You may specify a local pathname or an NFS network name; you may not use a UNC name or virtual drive mapping.

4. Each time a new federated database is registered as a data source, copy its boot file into the boot file directory (see step 2).

(Optional) Changing the Log Directory for the ODBC Server

At installation, the Objectivity / SQL++ ODBC server is configured to write log files to the location specified by the `temp` environment variable. To specify a different location:

1. Log on as administrator or as a user with equivalent privileges.
2. Click **Start** and point to **Programs**. In the Objectivity submenu, select **Objectivity Network Services**.
3. Select the Objectivity / SQL++ ODBC server and click the **Configure** button.
4. Enter or select the pathname of the log directory and click **OK**.

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, in a command window, enter:

   ```
   xcopy installDir\samples\sql\ooisql c:\isqldemo
   c:
   cd \isqldemo
   ```

2. Edit `makefile` in the directory you just created (in this example, `c:\isqldemo`):
   - Set `INSTALL_DIR` to be the location of the Objectivity / DB installation directory—for example:

     ```
     INSTALL_DIR = c:\obj60
     ```

   - Set `SQL_ROOT` to be the location of the Objectivity / SQL++ installation directory (which is the same as `INSTALL_DIR`)—for example:

     ```
     SQL_ROOT = c:\obj60
     ```

   - Set `LS_HOST` to be the name of the host running the lock server—for example:

     ```
     LS_HOST = myLockServerHost
     ```

3. Check whether the lock server is running; start it, if necessary.

4. Build the executables and the demo database. At the command prompt, enter:
   `nmake`
5. Run the demo application. At the command prompt, enter:
   `demo`

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

```
echo "Creating the Objectivity/DB Database."
echo "Running OOISQL to create views."
ooisql -input views.sql -user systpe -passwd dummy DEMO
   > log 2>err
echo "Running OOISQL to test out various SQL statements."
ooisql -input test.sql -user systpe -passwd dummy DEMO
   > log 2>err
echo "Comparing the results."
fc log OOISQLOK > diffs 2>err
echo Listing differences (if any). Null difference indicates -
   test passed.
echo Test completed
echo Listing errors (if any). Null listing indicated - test
   passed
echo End of error list
```

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++ programming 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 4 beginning on page 50 (*do not perform step 5*).
   - If you have already run the Interactive SQL++ demo application:
     - Change to the demo directory you used (for example, `c:\isqldemo`)
     - Check whether the lock server is running; start it, if necessary
     - Clean up the directory by entering `nmake clean`
     - Re-create the federated database and application by entering `nmake`

2. Copy the Objectivity/SQL++ interface demo directory to a new location and change your working directory to this location. For example, in a command window, enter:

   ```
   xcopy installDir\samples\sql\ooapi c:\apidemo
   c:
   cd \apidemo
   ```

3. Edit `makefile` in the directory you just created (in this example, `c:\apidemo`):
   - Set `INSTALL_DIR` to be the location of the Objectivity/DB installation directory—for example:
     ```
     INSTALL_DIR = c:\obj60
     ```
   - Set `SQL_ROOT` to be the location of the Objectivity/SQL++ installation directory (which may be different from `INSTALL_DIR`)—for example:
     ```
     SQL_ROOT = c:\objsql
     ```
   - Set `ooisqldemo` to be the location of the Interactive SQL++ demo that you built in step 1—for example:
     ```
     ooisqldemo = c:\isqldemo
     ```
4. Build and run the demo application. At the command prompt, enter:
   ```
   nmake
   ```

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.
No errors.
```

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 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 entire Interactive SQL++ demo (steps 1 through 5 beginning on page 50) to create and populate the demo federated database to be browsed. Be sure to leave the lock server running.

2. Check whether the Objectivity/SQL++ ODBC server is running; start it if necessary:
   a. Log in as administrator or as a user with equivalent privileges.
   b. Click the Windows **Start** button and point to the **Programs** menu. In the Objectivity submenu, select **Objectivity Network Services**.
   c. Select the Objectivity/SQL++ ODBC server and click the **Start** button.
3. If you have not already done so, create a boot file directory and configure the ODBC server accordingly (see "Setting Up a Boot File Directory" on page 49).
4. Copy the boot file `DEMO` from the Interactive SQL++ demo directory into the boot file directory. For example, enter:
   ```
   copy c:\isqldemo\DEMO c:\oodata
   ```
5. If the Objectivity/SQL++ ODBC Driver demo is to be performed by another user, give that user the TCP/IP name of the ODBC server host and the boot filename (`DEMO`).
6. Grant all access rights to all users for the tables in the demo federated database. If you omit this step, only the Objectivity/SQL++ database administrator (`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 by following the steps in "Testing Objectivity/ODBC" on page 60. The demo can be repeated as long as the lock server and the Objectivity/SQL++ ODBC server are both running.

8

Objectivity/SQL++ ODBC Driver Installation

This chapter describes the requirements and steps for installing the Objectivity/SQL++ ODBC Driver (Objectivity/ODBC) on a Windows platform. Objectivity/ODBC enables ODBC-compliant client applications, such as PowerBuilder, to access an Objectivity/SQL++ ODBC server, which in turn accesses Objectivity/DB federated databases. See Chapter 7, "Objectivity/SQL++ Installation," for information about the Objectivity/SQL++ ODBC server.

System Requirements

You can install Objectivity/ODBC on the Windows platforms listed in Table 8-1.

Table 8-1: Supported Windows Platforms for Objectivity/ODBC

Hardware	Operating System	Abbreviation
Intel Pentium or greater	Windows 98	Windows 98
	Windows NT Workstation Windows NT Server	Windows NT
	Windows 2000 Professional Windows 2000 Server	Windows 2000

Software Requirements

Objectivity / ODBC requires that the following software be installed on each computer that is to run an ODBC-compliant client application:

- Winsock-compatible TCP / IP software
- Microsoft ODBC Administrator 2.x and up

Microsoft TCP / IP and Microsoft ODBC Administrator are included with Windows operating systems.

In addition, Objectivity / ODBC requires that an Objectivity / SQL++ ODBC server be installed in the same network.

Installing Objectivity/ODBC

To install Objectivity / ODBC:

1. Verify that required software has been completely and correctly installed. See "Software Requirements" on this page.
2. Insert the **Objectivity/SQL++ ODBC Driver** diskette into your machine's 3.5-inch floppy disk drive.
3. Run `a:\setup` , where `a:` is the drive containing the diskette.
4. Follow the prompts to complete the installation.
5. Add the desired Objectivity / DB federated databases as data sources. Follow the steps in "Adding Objectivity / DB Data Sources" on page 57.
6. Configure TCP / IP to enable Objectivity / ODBC to communicate with an Objectivity / SQL++ ODBC server. Follow the steps in "Configuring the Network" on page 59.
7. (Optional) Test Objectivity / ODBC with an Objectivity / SQL++ ODBC server by browsing the provided demo federated database (see "Testing Objectivity / ODBC" on page 60).

What Objectivity/ODBC Setup Does

The Objectivity / ODBC setup program inserts an entry for Objectivity / ODBC in your machine's `odbcinst.ini` file. This entry makes the driver available to the Microsoft ODBC Administrator.

Adding Objectivity/DB Data Sources

You enable ODBC-compliant client applications to access an Objectivity / DB federated database by adding a *data source*—the data to be accessed and the means of access (typically host and network information)—that identifies the federated database in a unique combination of two parameters: a boot file and a host running a particular Objectivity / SQL++ ODBC server.

Before you register a federated database as a data source:

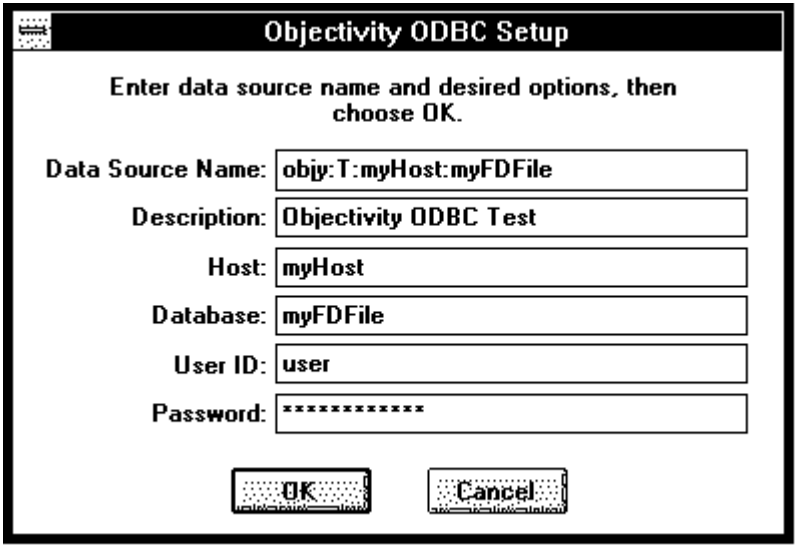
- Identify the Objectivity / SQL++ ODBC server that will access the federated database, and obtain the TCP / IP name of the host that runs this server.
- Obtain the simple name of the federated database's boot file (more specifically, the name of the copy that was placed in the boot file directory for the Objectivity / SQL++ ODBC server).
- Ensure that you have a valid user account on the host running the Objectivity / SQL++ ODBC server.

To register a federated database as a data source:

1. On the host where you installed Objectivity / ODBC (the driver), open the Control Panel and double-click **ODBC Data Source**.
2. In ODBC Data Source Administrator, click **System DSN**.
3. Click **Add**, select **Objectivity/SQL++ ODBC Driver (32-bit)** in the Create New Data Sources dialog, and click **Finish**.
4. In the resulting dialog, fill in the following fields and then click **OK**:

Data Source Name	A string that uniquely identifies the data source. This string will appear in the list of data sources in the connection dialog. This string must start with the case-sensitive prefix `obj y:T:` and conventionally includes the information you enter in the Host and Database fields—for example: `obj y:T:myHost:myFDFile`. The string may not exceed 32 characters.
Description	(Optional) Description of the data source; used for documentation purposes only.
Host	TCP/IP name of the Objectivity/SQL++ ODBC server host.

Database	Name of the boot file for the federated database (more specifically, the name of the copy that resides in the boot file directory). This name must not exceed 10 characters (including any filename extension) and may not contain spaces. Do not include the directory path for this file; the ODBC server is preconfigured to find the boot file directory.
User ID	The username of your Objectivity/SQL++ account. This is normally your login account on the ODBC server host, provided that this account has been granted access rights to tables in the federated database (see your Objectivity/SQL++ database administrator). The Objectivity/SQL++ database administrator account (`systpe`) has access to all tables.
Password	Password for the account you entered in the User ID field. The password you enter is *not* encoded before it is sent across the network.



5. If you want to add other data sources, repeat steps 3 and 4.
6. When you are finished adding data sources, click **Exit**.

Configuring the Network

Identifying the ODBC Server's Host to TCP/IP

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

If your site uses a TCP/IP `hosts` file, you must make sure it contains an entry for the hostname of the Objectivity/SQL++ ODBC server to which your client application is to connect. To do this:

1. Open the TCP/IP `hosts` file on the host where you installed Objectivity/ODBC (the driver). The location of this file depends on the TCP/IP vendor (see Table 8-2 for the Microsoft TCP/IP file locations).

Table 8-2: Location of Microsoft TCP/IP Hosts File

Operating System	Hosts File
Windows NT and Windows 2000	`C:\winnt\system32\drivers\etc\hosts`
Windows 98	`C:\windows\hosts`

2. Add an entry with the following format, if such an entry does not already exist:
   `internetAddress` *`hostName`*
   where

`internetAddress`	IP address for the Objectivity/SQL++ ODBC server host
`hostName`	The name you will use to refer to the Objectivity/SQL++ ODBC server host

EXAMPLE The following entry from a TCP/IP `hosts` file includes both a hostname and a domain name:

```
192.42.242.23            objy23 objy23.objy.com
```

Specifying the ODBC Server's Port Number

As installed, the Objectivity/SQL++ ODBC server and the Objectivity/SQL++ ODBC Driver communicate through a default TCP/IP port. If the ODBC server has been assigned a nondefault port number (for example, due to a port conflict),

you must register the new port number with TCP/IP on each Objectivity/ODBC (driver) host.

To register the ODBC server's port number with TCP/IP:

1. Find the TCP port assigned to the `oosqlnw` service on the host running the Objectivity/SQL++ ODBC server. For example, use the Objectivity Network Services tool on the ODBC server host.
2. On the host where you installed Objectivity/ODBC (the driver), open the TCP/IP `services` file. The location of this file depends on the TCP/IP vendor (see Table 8-3 for the Microsoft TCP/IP file locations).

Table 8-3: Location of Microsoft TCP/IP Services File

Operating System	Services File
Windows NT and Windows 2000	`C:\winnt\system32\drivers\etc\services`
Windows 98	`C:\windows\services`

3. Add the following entry to the TCP/IP `services` file, if such an entry does not already exist:

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

   where

   `portNumber` TCP port number

Testing Objectivity/ODBC

You can verify the correct operation of Objectivity/ODBC in combination with an Objectivity/SQL++ ODBC server by successfully browsing a demo federated database through Microsoft Access. The demo federated database is provided with Objectivity/SQL++ on the server host.

To test Objectivity/ODBC installation:

1. Verify that the demo federated database and the Objectivity/SQL++ ODBC server have been set up for this test:
   - If your ODBC server runs on Windows NT, see "Preparing the ODBC Server for Testing" in Chapter 7 in this book.
   - If your ODBC server runs on UNIX, see "Preparing the ODBC Server for Testing" in Chapter 7 in *Installation and Platform Notes for UNIX*.

2. On the Objectivity / ODBC host, register the federated database from the Interactive SQL++ demo as the data source. Perform the steps in "Adding Objectivity / DB Data Sources" on page 57 using the following information:

Data Source Name	`obj y:T:`*`hostName`*`:DEMO`
Host	*`hostName`* (name of the host running your Objectivity/SQL++ ODBC server).
Database	`DEMO` (name of the boot file for the demo federated database).
User ID	A username that has been granted access rights for the tables in the demo federated database. By default, this is the Objectivity/SQL++ database administrator account (`systpe`) unless the Objectivity/SQL++ database administrator has granted access to other user accounts on the ODBC server host.
Password	Password for the account you entered in the User ID field.

3. Open Microsoft Access.
4. Create a new blank database.
5. Select **File > Get External Data > Link Tables**.
6. In Link, choose **ODBC Databases()** from the Files of Type list.
7. In Select Data Source, click **Machine Data Source** and choose the data source you registered in step 2 (`Obj y:T:`*`hostName`*`:DEMO`); click **OK**.
8. In Link Tables, select the table `systpe.component` and click **OK**.
9. In the dialog, select `name` as the unique record identifier, and browse the data.

Successfully browsing the data indicates that Objectivity / ODBC is installed correctly.

9

Objectivity/FTO Installation

This chapter describes the requirements and steps for installing Objectivity / DB Fault Tolerant Option (Objectivity / FTO) on a Windows 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 the Windows platforms listed in Table 1-1 on page 11.

Software Requirements

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

- Objectivity / DB (see Chapter 1)

Installing Objectivity/FTO

To install Objectivity / FTO:

1. Verify that all required software has been completely and correctly installed (see “Software Requirements” above).
2. Place the Objectivity CD in your CD-ROM drive. The setup program for installing Objectivity products will start automatically.

   If you need to start the setup program explicitly, display the CD-ROM drive and double-click `setup.exe` on the CD.

3. Click **Modify** and then click **Next** to display a list of Objectivity products; a check mark ✓ next to a product indicates that the product is already installed.
4. Select **Objectivity/DB Fault Tolerant Option**. If a required product is not yet installed, it is selected and installed automatically.

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

5. Click **Next** and follow the prompts to complete the installation.
6. If you also have Objectivity / Smalltalk for VisualWorks installed, follow the steps in "Setting Up Objectivity / Smalltalk for VisualWorks" on page 65.
7. Read:
   - *Objectivity Release Notes* for new and changed features. (Click **Start** and point to **Programs**. In the Objectivity submenu, click **About This Release**.)
   - 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.

What Objectivity/FTO Setup Does

For Objectivity / FTO, the setup program installs files in subdirectories of the Objectivity / DB installation directory *`installDir`*, as shown in Table 9-1.

Table 9-1: Objectivity/FTO Release Files in *`installDir`*

Subdirectory	Contains
`bin`	Executables for Objectivity/FTO tools (see the Objectivity/FTO and Objectivity/DRO book).
	Objectivity/DB option-enabling DLL `oochk`*`xx`*`.dll` (see "Libraries for Dynamic Linking" on page 74). This DLL replaces the version that is installed with Objectivity/DB.
`doc`	PDF file for the *Objectivity/FTO and Objectivity/DRO* online book.
`etc\smalltlk`	Files providing the Smalltalk programming interface to Objectivity/FTO.

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\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*\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.

10

Objectivity/DRO Installation

This chapter describes the requirements and steps for installing Objectivity / DB Data Replication Option (Objectivity / DRO) on a Windows 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 the Windows platforms listed in Table 1-1 on page 11.

Software Requirements

Objectivity / DRO requires that the following software be installed on your computer:

- Objectivity / DB (see Chapter 1)

  The Advanced Multithreaded Server (AMS) must be installed on every host that is to contain a replicated database (see “Verifying Objectivity Server Status” on page 16).
- Objectivity / FTO (see Chapter 9)

Installing Objectivity/DRO

To install Objectivity / DRO:

1. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 67).
2. Place the Objectivity CD in your CD-ROM drive. The setup program for installing Objectivity products will start automatically.

   If you need to start the setup program explicitly, display the CD-ROM drive and double-click `setup.exe` on the CD.
3. Click **Modify** and then click **Next** to display a list of Objectivity products; a check mark ✓ next to a product indicates that the product is already installed.
4. Select **Objectivity/DB Data Replication Option**. If a required product is not yet installed, it is selected and installed automatically.

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

---

5. Click **Next** and follow the prompts to complete the installation.
6. If you also have Objectivity / Smalltalk for VisualWorks installed, follow the steps in “Setting Up Objectivity / Smalltalk for VisualWorks” on page 69.
7. Verify that AMS is installed and running on every host that is to contain a replicated database (see “Verifying Objectivity Server Status” on page 16). Although AMS need not be running when you create an original database image, you must start AMS before you can create additional database images.
8. Read:
   - *Objectivity Release Notes* for new and changed features. (Click **Start** and point to **Programs**. In the Objectivity submenu, click **About This Release**.)
   - 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.

What Objectivity/DRO Setup Does

For Objectivity / DRO, the setup program installs files in subdirectories of the Objectivity / DB installation directory *`installDir`*, as shown in Table 10-1.

Table 10-1: Objectivity/DRO Release Files in *`installDir`*

Subdirectory	Contains
`bin`	Executables for Objectivity/DRO tools.
	Objectivity/DB option-enabling DLL `oochk`*`xx`*`.dll` (see "Libraries for Dynamic Linking" on page 74). This DLL replaces the version installed with Objectivity/DB.
`etc\smalltlk`	Files providing the Smalltalk programming interface to Objectivity/DRO.

NOTE The online book that describes Objectivity / DRO is installed with Objectivity / FTO.

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`*`\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`*`\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.

11

Objectivity/IPLS Installation

This chapter describes the requirements and steps for installing Objectivity / DB In-Process Lock Server Option (Objectivity / IPLS) on a Windows 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 the Windows platforms listed in Table 1-1 on page 11.

Software Requirements

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

- Objectivity / DB (see Chapter 1)

Installing Objectivity/IPLS

To install Objectivity / IPLS:

1. Verify that all required software has been completely and correctly installed (see “Software Requirements” on page 71).
2. Place the Objectivity CD in your CD-ROM drive. The setup program for installing Objectivity products will start automatically.

   If you need to start the setup program explicitly, display the CD-ROM drive and double-click `setup.exe` on the CD.
3. Click **Modify** and then click **Next** to display a list of Objectivity products; a check mark ✓ next to a product indicates that the product is already installed.

4. Select **Objectivity/DB In-Process Lock Server Option**. If a required product is not yet installed, it is selected and installed automatically.

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

5. Click **Next** and follow the prompts to complete the installation.
6. Read:
   - *Objectivity Release Notes* for new and changed features. (Click **Start** and point to **Programs**. In the Objectivity submenu, click **About This Release**.)
   - 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.

What Objectivity/IPLS Setup Does

For Objectivity / IPLS, the setup program installs files in subdirectories of the Objectivity / DB installation directory *`installDir`*, as shown in Table 11-1.

Table 11-1: Objectivity/IPLS Release Files in *`installDir`*

Subdirectory	Contains
`bin`	Dynamic link libraries for Objectivity/IPLS

Using Objectivity/IPLS

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

No extra steps are required to compile and link an IPLS application. The appropriate Objectivity / IPLS DLL is loaded automatically at runtime when the application starts the in-process lock server. The Objectivity / IPLS DLLs are listed in Table A-13 on page 81.

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 Windows platforms. You should use this appendix in conjunction with the Objectivity/C++ and Objectivity/DDL books.

This appendix provides information about:

- Linking applications to Objectivity/DB
- Linking applications to additional options and features
- User-created DLLs that export Objectivity/DB persistent data
- Objectivity/C++ application programming issues
- Building Objectivity/C++ applications
- Memory-checking software
- Debugging Objectivity/C++ applications
- Where to find sample Objectivity/C++ applications

Linking Applications to Objectivity/DB

On Windows, you dynamically link your C++ applications to Objectivity/DB. The appropriate Objectivity/DB libraries are normally linked automatically, so you do not need to specify libraries explicitly to the linker (see "Automatic Linking of Objectivity/DB Libraries" on page 75).

The following subsections describe the libraries required for linking to Objectivity/DB, guidelines for combining Objectivity/DB libraries with runtime libraries from other vendors, and what you need to know about automatic linking.

Libraries for Dynamic Linking

Table A-1 lists the import libraries for linking applications dynamically to Objectivity / DB.

Table A-1: Objectivity/DB Dynamic Link Import Libraries

Library File	Description
`oodbi.lib`	Objectivity/DB import library
`oodbid.lib`	Debug version of the Objectivity/DB import library

Table A-2 lists the dynamic link libraries (DLLs) that must be available at runtime to applications that are linked dynamically to Objectivity / DB.

Table A-2: Objectivity/DB Dynamic Link Libraries

DLL File[a]	Description
`oochk`*`xx`*`.dll`	Objectivity/DB option-enabling DLL (enables features for licensed options in the Objectivity/DB DLL)
`oodb`*`xx`*`.dll`	Objectivity/DB DLL (corresponds to the `oodbi.lib` import library)
`oodb`*`xx`*`d.dll`	Debug version of the Objectivity/DB DLL (corresponds to the `oodbid.lib` import library)

a. The digits *xx* in a DLL name correspond to the current Objectivity/DB release.

Release applications link to the Objectivity / DB import library (`oodbi.lib`) and use the corresponding Objectivity / DB DLL (`oodb`*`xx`*`.dll`). In contrast, debug applications link to the debug-compatible import library (`oodbid.lib`) and use the corresponding DLL (`oodb`*`xx`*`d.dll`). This is necessary because the Visual C++ debug runtime libraries redefine basic memory allocation routines. Failure to link to the correct library may result in a runtime exception, left-over journal files, or data corruption.

All applications (release and debug) use `oochk`*`xx`*`.dll`. Three different versions of `oochk`*`xx`*`.dll` exist—one for each product (Objectivity / DB, Objectivity / FTO, and Objectivity / DRO). Each version enables the appropriate features in the Objectivity / DB release and debug libraries (see Table A-3).

Table A-3: Versions of `oochk`*xx*`.dll`

Version Installed With	Enables Features for
Objectivity/DB	Objectivity/DB
Objectivity/FTO	Objectivity/DB and Objectivity/FTO
Objectivity/DRO	Objectivity/DB, Objectivity/FTO, and Objectivity/DRO

Important: To guarantee orderly process shutdown of a multithreaded application (including unloading Objectivity DLLs in the correct order), you must call the `ooExitCleanup` function before returning from `main()` or calling `exit`. See the Objectivity / C++ programmer's reference.

Automatic Linking of Objectivity/DB Libraries

When you build your application, the appropriate Objectivity / DB libraries are linked automatically, so you do not need to specify them explicitly to the linker. This is because the `oo.h` include file contains pragmas that direct the linker to link to the correct Objectivity / DB libraries. The pragmas in the `oo.h` file are similar to the pragmas defined by Microsoft in the `afx.h` include file. (Note that `oo.h` is included in your application automatically when you run the DDL processor.)

Table A-4 shows the runtime libraries that are selected automatically when you link a release or debug version of an application.

Table A-4: Compatible Runtime Libraries Selected for Linking

Application Type	Visual C++ Multithreaded Runtime Library	Objectivity/DB Library
Release version	`msvcrt.lib + $(winlibs)`	`oodbi.lib`
Debug version	`msvcrtd.lib + $(winlibs)`	`oodbid.lib`

The following subsections describe how automatic linking is triggered.

If You Use the Visual C++ IDE

If you develop using the Visual C++ Integrated Development Environment (IDE), your project selects the Visual C++ runtime library, the Objectivity / DB library, and other libraries based on the value you set for either of the link options listed in Table A-5.

Table A-5: Link Options Controlling Library Selection

Visual C++ IDE Option	Value
Project > Settings > C/C++ > Category:Code Generation > Use Run-Time Library	**Multithreaded DLL** or **Debug Multithreaded DLL**
Project > Settings > General > Microsoft Foundation Classes	**Use MFC in a Shared DLL**

You should set one of these options, but you do not need to set both; the MFC link option automatically sets the Visual C++ runtime link option. Regardless of the option you choose, you must set it to select a DLL (either for a release or a debug version). You can also toggle between release and debug versions using the **Project > Settings** option.

If You Use nmake

If you develop using `nmake.exe` from the command line, the appropriate Objectivity / DB library is selected by the option you give to the `cl.exe` command in the makefile, as shown in Table A-6.

Table A-6: Command LIne Options Controlling Library Selection

Option Given to `cl.exe`	Visual C++ Multithreaded Runtime Library	Objectivity/DB Library
`-MD`	Multithreaded DLL	`oodbi.lib`
`-MDd`	Debug multithreaded DLL	`oodbid.lib`

You must specify one of the options shown in Table A-6. For an example of the makefile options, inspect the following sample file:

`installDir``\samples\cppdll\makefile`

Conflicting Symbols Warning

For most projects, automatic linking of Objectivity / DB libraries is sufficient. However, occasionally you may receive a warning that symbols in default libraries conflict. To correct this, you must specify to ignore the standard Visual C++ runtime library. In the Visual C++ IDE, you:

1. Select **Project > Settings > Link > Category:Input**
2. Enter `msvcrt.lib` in **Ignore libraries**.

Linking Explicitly

In general, you should take advantage of automatic linking whenever possible. If you prefer to specify Objectivity / DB libraries to the linker explicitly, you *must* preserve the following link order:

1. User-created object files and libraries
2. Objectivity / DB library
3. Microsoft Foundation Classes (MFC) libraries
4. Visual C++ runtime library

When specifying libraries explicitly through the Visual C++ IDE:

1. Select **Project > Settings > Link > Category:Input**
2. Enter the desired libraries in **Object/library modules**, preserving the link order shown above.
3. Click **Ignore all default libraries**.

Compatibility With Other Runtime Libraries

Objectivity / DB libraries link only to the dynamic multithreaded Visual C++ runtime libraries. If you use libraries from other vendors, these libraries must be compatible with the same version of the multithreaded Visual C++ runtime libraries. Using incompatible libraries may result in either a link-time failure or data corruption.

Because you link dynamically to Objectivity / DB, you should also link dynamically to any third-party libraries. Mixing static and dynamic link libraries may result in either a link-time failure or data corruption.

Linking to Additional Objectivity Products and Features

If an application uses Objectivity / C++ persistent collections or the C++ programming interfaces of other Objectivity products, the application must include the header files that are specified in the documentation for the feature or product. When a release or debug application includes such header files, the appropriate release or debug import libraries are linked automatically. In all cases, the corresponding dynamic link libraries (DLLs) must be available at runtime.

Linking to Objectivity/C++ Persistent Collections

Table A-7 lists the import libraries for the Objectivity / C++ persistent collections feature. The release or debug import library is linked automatically when you include the persistent collections header file.

Table A-7: Objectivity/C++ Persistent Collections Dynamic Link Import Libraries

Library File	Description
`ooco.lib`	Multithreaded import library for Objectivity/C++ persistent collections
`oocod.lib`	Debug version of the multithreaded import library for Objectivity/C++ persistent collections

Table A-8 lists the DLLs that must be available at runtime to applications that are linked dynamically to Objectivity / C++ persistent collections.

Table A-8: Objectivity/C++ Persistent Collections Dynamic Link Libraries

DLL File[a]	Description
`ooco`*`xx`*`.dll`	DLL for Objectivity/C++ persistent collections
`ooco`*`xx`*`d.dll`	Debug version of the DLL for Objectivity/C++ persistent collections

a. The digits *xx* in a DLL name correspond to the current Objectivity/DB release.

Linking to Objectivity/C++ STL

Table A-9 lists the import libraries for Objectivity/C++ STL, which adds persistence to ObjectSpace Standards<Toolkit> STL classes. Including the Objectivity/C++ STL header files automatically links the appropriate import libraries for *both* the ObjectSpace STL and Objectivity/C++ STL.

Table A-9: Objectivity/C++ STL Dynamic Link Import Libraries

Library File	Description
`std-vc-mt.lib`	ObjectSpace STL multithreaded import library
`std-vc-debug-mt.lib`	Debug version of the ObjectSpace STL multithreaded import library
`objystl-vc-mt.lib`	Objectivity/C++ STL multithreaded import library
`objystl-vc-debug-mt.lib`	Debug version of the multithreaded import library for Objectivity/C++ STL

Table A-10 lists the DLLs that must be available at runtime to applications that are linked dynamically to Objectivity/C++ STL.

Table A-10: Objectivity/C++ STL Dynamic Link Libraries

DLL File	Description
`std-vc-mt.dll`	DLL for ObjectSpace STL
`std-vc-debug-mt.dll`	Debug version of the DLL for ObjectSpace STL
`objystl-vc-mt.dll`	DLL for Objectivity/C++ STL
`objystl-vc-debug-mt.dll`	Debug version of the DLL for Objectivity/C++ STL

Linking to Objectivity/C++ Active Schema

Table A-11 lists the import libraries for Objectivity / C++ Active Schema (Objectivity / AS), which enables applications to dynamically read and modify a federated database schema. The release or debug import library is linked automatically when you include the Objectivity / AS header file.

Objectivity / AS depends on ObjectSpace Standards<Toolkit> STL classes, so ObjectSpace STL import libraries are distributed with Objectivity / AS and are linked automatically when you include the Objectivity / AS header file.

Table A-11: Objectivity/AS Dynamic Link Import Libraries

Library File	Description
`ooas.lib`	Objectivity/AS multithreaded import library
`ooasd.lib`	Debug version of the Objectivity/AS multithreaded import library
`std-vc-mt.lib`	ObjectSpace STL multithreaded import library
`std-vc-debug-mt.lib`	Debug version of the ObjectSpace STL multithreaded import library

Table A-12 lists the DLLs that must be available at runtime to applications that are linked dynamically to Objectivity / AS.

Table A-12: Objectivity/AS Dynamic Link Libraries

DLL File[a]	Description
`ooas`*`xx`*`.dll`	Objectivity/AS DLL
`ooas`*`xx`*`d.dll`	Debug version of the Objectivity/AS DLL
`std-vc-mt.dll`	DLL for ObjectSpace STL
`std-vc-debug-mt.dll`	Debug version of the DLL for ObjectSpace STL

a. The digits *xx* in a DLL name correspond to the current Objectivity/DB release.

Automatic Loading of Objectivity/IPLS

An Objectivity / C++ application can start an in-process lock server without linking to any special import library. Instead, the appropriate Objectivity / IPLS dynamic link library is loaded automatically at runtime when the in-process lock server is started.

Table A-13 lists the DLLs that must be available at runtime to applications that use Objectivity / IPLS.

Table A-13: Objectivity/IPLS Dynamic Link Libraries

DLL File[a]	Description
`ooipls`*xx*`.dll`	DLL for Objectivity/IPLS
`ooipls`*xx*`d.dll`	Debug version of the Objectivity/IPLS DLL

a. The digits *xx* in a DLL name correspond to the current Objectivity/DB release.

Linking a Lock-Server Performance-Monitoring Program

Table A-14 lists the import 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). The release or debug import library is linked automatically when you include the `oolspm.h` header file.

Table A-14: Dynamic Link Import Libraries

Library File	Description
`oolspmi.lib`	Multithreaded import library for custom lock-server performance-monitoring programs
`oolspmid.lib`	Debug version of the multithreaded import library for custom lock-server performance-monitoring programs

Table A-15 lists the dynamic link libraries (DLLs) that must be available at runtime to lock-server performance-monitoring tools.

Table A-15: Dynamic Link Libraries

DLL File[a]	Description
`oolspm`*xx*`.dll`	DLL for custom lock-server performance-monitoring programs
`oolspm`*xx*`d.dll`	Debug version of the DLL for custom lock-server performance-monitoring programs.

a. The digits *xx* in a DLL name correspond to the current Objectivity/DB release.

User-Created DLLs and Objectivity/DB

You can create DLLs whose interfaces export Objectivity / DB persistent data. The following subsections show how to export Objectivity / DB persistent data from a user-created DLL, and provide guidelines for linking and loading these DLLs. These subsections assume that you already know how to create a DLL.

For a sample program, see the directory *installDir*`\samples\dll`.

Exporting Persistent Data From a User-Created DLL

This section describes how to export persistent data from a DLL. You must be familiar with the use of the Microsoft Extended Attribute Syntax `__declspec(dllexport)` and `__declspec(dllimport)`.

To export Objectivity / DB persistent data:

1. Choose a placeholder symbol name to represent the extended storage attribute—for example, `MY_STORAGE_SPECIFIER`.
2. Specify the extended storage attribute in the class definition of each class to be exported from the DLL. You can use either the placeholder symbol or the `__declspec(dllexport)` syntax.

   For example, assume you choose `MY_STORAGE_SPECIFIER` as the placeholder symbol and you want to export `MyClass` from your DLL. You specify the extended storage attribute in the definition of `MyClass` using either of the following alternatives:

   ```
   class MY_STORAGE_SPECIFIER MyClass : public ooObj {
   ```

   or

   ```
   class __declspec(dllexport) MyClass : public ooObj {
   ```
3. Run `ooddlx.exe`, using the `-storage_specifier` option to pass the placeholder symbol to it. For example:

   ```
   ooddlx -storage_specifier MY_STORAGE_SPECIFIER
   ```

   When run with this option, `ooddlx.exe`:

   - Directly passes any instance of the specified placeholder symbol to the output.
   - Converts all instances of `__declspec(dllexport)` to the specified placeholder symbol on output.

NOTE If you run `ooddlx.exe` without the `-storage_specifier` option, all instances of `__declspec(dllexport)` are passed directly through to the output without being translated.

---

4. Compile your DLL using the `-D` compiler option to replace the placeholder symbol with `__declspec(dllexport)`. For example:
   `-DMY_STORAGE_SPECIFIER=__declspec(dllexport)`
   An import library will be generated automatically when you link the DLL, so you may not need a `.def` file.
5. Compile the applications that use the DLL with the `-D` compiler option to replace the placeholder symbol with `__declspec(dllimport)`. For example:
   `-DMY_STORAGE_SPECIFIER=__declspec(dllimport)`

Linking User-Created DLLs to Objectivity/DB

Linking a user-created DLL to Objectivity / DB is similar to linking an application to Objectivity / DB. That is, the same considerations apply for combining Objectivity / DB libraries with other third-party libraries, and for automatic linking.

Incorporating a User-Created DLL

Linking an Application to an Import Library

You can incorporate a user-created DLL into an application by linking the application to an import library for the DLL.

Loading a User-Created DLL

You can incorporate a user-created DLL into an application by using the Win32 `LoadLibrary` function. This is an alternative to linking the application to an import library for the DLL. When you load a user-created DLL using the `LoadLibrary` function, you must also link the DLL and your application as stated in “Linking User-Created DLLs to Objectivity / DB” on page 83.

If you intend to use the Win32 `LoadLibrary` function to load a user-created DLL, you should not use the `__declspec(thread)` Visual C++ language extension on any data in that DLL. Using `__declspec(thread)` in this way will result in a runtime exception. This is a known Microsoft Visual C++ restriction.

Application Programming Issues

Using the Microsoft Foundation Classes

Objectivity / DB is compatible with the Microsoft Foundation Classes (MFC) as long as the two class hierarchies are kept separate. Mixing the class hierarchies (either by inheritance or by object inclusion) is not supported. The reason for this

restriction is that the class library implementation may rely on in-memory pointers, which become invalid once the object is written to disk.

The majority of MFC classes deal with GUI objects for which persistence makes little sense. You may be tempted to use the MFC collection classes in your persistent classes, but for the reason stated above the collections classes will not work as expected and you should avoid using them in your persistent classes (consider using Objectivity / C++ STL collection classes instead). Furthermore, you should not include MFC header files in your DDL schema files.

An example of integrating Objectivity / DB within an MFC application framework is provided in the *installDir*`\samples\mfclom` directory. See the `readme.txt` file located in that directory for more information.

Signal Handling

The Objectivity / DB predefined signal handler catches the following signals on Windows NT and Windows 2000:

- `SIGABRT`
- `SIGFPE`
- `SIGILL`
- `SIGSEGV`
- `SIGTERM`

Objectivity / DB does not catch the following signals and Windows events:

- `SIGINT`
- `SIGBREAK`
- The `ExitProcess` function
- Any exceptions that do not map directly to the above signals
- Stack overflow

A new thread is created when a user triggers a `SIGINT` event (from a Control-c) or a `SIGBREAK` event (from a Control-Break).

In general, if your application encounters one of the events that Objectivity / DB does not catch, you should recover your transactions using the `oocleanup.exe` tool described in the Objectivity / DB administration book.

Handling Microsoft Visual C++ Name Decoration

The Microsoft Visual C++ compiler uses its own algorithm to decorate (mangle) the names of functions and variables. The algorithm is different from that of `cfront` or other C++ compilers. If you encounter linking problems in your code because an external declaration is not found, the compiler may have decorated the function or variable name.

Also note that the Microsoft C++ compiler decorates the names of global variables, whereas most UNIX C++ compilers do not. If you have global variables that are shared between C++ and C modules, make sure that you declare and define these variables as `extern "C"` in your C++ code.

Building Applications

nmake-Based Development

You can develop Objectivity/C++ applications using `nmake`, a utility provided by Visual C++ that is very similar to `make` on UNIX. In `nmake`-based development, you build applications by setting up and using makefiles.

To build an Objectivity/C++ application on Windows NT or Windows 2000 using `nmake`, you should use a makefile that includes the file `win32.mak` (provided by Microsoft). Procedures for creating makefiles are contained in the Microsoft `nmake` documentation.

Objectivity/C++ provides sample applications and makefiles to demonstrate how to develop with `nmake`. A sample makefile that uses the Objectivity/DB dynamic library is in *`installDir`*`\samples\cppdll`. You can use this makefile as a template for incorporating Objectivity/DB into your own makefiles.

Visual C++ IDE-Based Development

You can develop Objectivity/C++ applications within the Visual C++ Integrated Development Environment (IDE). To do this, you must first set up the Visual C++ IDE to work with Objectivity/C++ (see “Setting Up the Visual C++ IDE” on page 27). Then you set up each of your own projects by:

1. “Creating a Custom Makefile for Your Project” on page 85
2. “Defining Your Project” on page 87
3. “Defining Custom Build Rules for DDL Files” on page 87
4. “Setting Link Options” on page 88
5. “Building Your Application” on page 88

Creating a Custom Makefile for Your Project

For each project, you must create a custom makefile called `makefile.mvc` that enables the Visual C++ IDE to create a federated database and run the DDL processor. This custom makefile is also required for supporting any Objectivity-specific actions you may have defined on the Visual C++ IDE **Tools** menu.

To create the custom makefile for a project, you:

1. Copy the files `makefile.mvc` and `makefile` from any sample Objectivity/C++ application into your project directory. For example, enter:
   ```
   copy installDir\samples\cppdll\makefile.mvc
   projectDir\makefile.mvc
   copy installDir\samples\cppdll\makefile
   projectDir\makefile
   ```
2. Edit your copy of `makefile` to set the makefile variables as appropriate to your application and the federated database to be created. Be sure to consider the variables listed in Table A-16. For details about creating a federated database, see `oonewfd` in the Objectivity/DB administration book.

Table A-16: Makefile Variables

Set This Variable	To Specify This Value
`BOOT_FILE`	Pathname of the boot file that is to represent the federated database.
`DDL_FILES`	List of the DDL files to be converted by the DDL processor into your database schema. Separate the filenames with spaces.
`FD_NUMBER`	Unique federated database identifier (a number from 1 to 32,000).
`FDB_FILE`	Filename of the federated database to be created.
`FILE_DIR`	Path of the directory in which the federated database will be created.
`FILE_HOST`	Name of the host on which the federated database will be created.
`LOCK_SERVER_HOST`	Name of the host running the lock server for your federated database. Set this variable to the TCP/IP name `localhost` if no lock server will be used.
`PAGE_SIZE`	Size of the unit of storage transfer between memory and disk.
`OBJY_ROOT_DIR`	Pathname of your Objectivity/DB installation directory.

Defining Your Project

You may use an existing project or you may define a new one for your Objectivity/C++ application. To define a new project file:

1. Choose **File > New**.
2. In the New dialog, select the **Projects** tab.
3. On the Projects page:
   a. Enter the project name (this also names the .exe file) and location.
   b. Select the appropriate project type from the list.
   c. Click **OK**.
4. Choose **Project > Add to Project > Files** and specify the names of all the source files.

Defining Custom Build Rules for DDL Files

You can set up your Visual C++ project so that the DDL processor is automatically invoked by the **Build > Build** or **Build > Rebuild All** command whenever necessary. To do this:

1. If you have not already done so, create the project's `makefile.mvc` file, as described in "Creating a Custom Makefile for Your Project" on page 85.
2. Select **Project > Add to Project > Files** and add your DDL files and the accompanying generated files to the project. (See the Objectivity/C++ Data Definition Language book for more information about DDL files and the files that the DDL processor generates from them.)
3. Select **Project > Settings > Custom Build**.
4. In the Settings For pulldown list, choose **All Configurations**, then select all the DDL files in the project. (Hint: Hold the Control key while selecting the files).
5. Specify the following command in the Commands list:
   ```
   nmake -f makefile.mvc ddl_build
   ```
6. Add the generated files to the Outputs list:
   ```
   $(InputName)_ref.h
   $(InputName)_ddl.cpp
   $(InputName).h
   ```

Setting Link Options

You can set Visual C++ IDE options as shown in the following steps to trigger automatic linking of Objectivity / DB libraries (see also “Automatic Linking of Objectivity / DB Libraries” on page 75):

- To link dynamic using the MFC:
  a. Select **Project > Settings > General**
  b. Choose the following value in the Microsoft Foundation Classes list: **Use MFC in a Shared DLL**
- To link dynamic without using the MFC:
  a. Select **Project > Settings > C/C++ > Category: Code Generation**
  b. Choose the following value in the Use run-time library list: **Multithreaded DLL**

Building Your Application

After you have completed the steps in the preceding subsections, you can build your application. To do this:

1. Choose **File > Open** and select the appropriate project file.
2. Choose **Build > Rebuild All**.

   Because of the custom makefile (`makefile.mvc`) and the custom build rules you defined, this step automatically creates a federated database, runs the DDL processor, and builds the application.

Using Memory-Checking Software

Several third-party tools and libraries track an application’s use of memory and report memory leaks. Unfortunately, the algorithm used by the majority of these tools and libraries, including the Visual C++ debug runtime library, considers any memory not deallocated at process `exit` to be a leak, which is incorrect. Many programs rely on process `exit` to free allocated memory and return it to the operating system. A true leak is a block of allocated memory that no longer has a program reference to it (the block has been lost).

Objectivity / DB allocates and keeps track of memory during the lifetime of the process, and uses the process `exit` to free this memory. Objectivity / DB has been fully tested with sophisticated memory diagnostic tools and found to be leak free. However, some tools will report the memory allocated by Objectivity / DB as a leak.

Debugging an Application

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

- Use Objectivity / DB tools for viewing and changing federated databases (see Chapter 5, "Debugging a Federated Database," of 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 next two subsections. The remaining subsections describe how to view persistent objects from the Visual C++ debugger.

Preparing to Debug an Application (Visual C++ IDE)

You can use Visual C++ to debug an Objectivity / C++ application built inside the Visual C++ IDE. To do this:

1. Select **Project > Settings** and, in the Settings For pulldown list, choose **Win32 Debug**.
2. Select **Project > Settings > C/C++ > Category:General** and choose **Program Database** from the Debug Info list.
3. Select **Project > Settings > Link > Category:Debug** and choose **Debug info** from the Debug Info options.
4. Link your application to the debug-compatible Objectivity / DB import library (see "Automatic Linking of Objectivity / DB Libraries" on page 75).

Preparing to Debug an Application (nmake)

You can use the Visual C++ debugger to debug an Objectivity / C++ application that was built using `nmake` from the command line. To do this:

1. Build the application using the appropriate predefined compile `$(cdebug)` and link `$(ldebug)` options provided in `win32.mak`. The following sample file demonstrates the correct use of these options:

   `installDir\samples\cppstat\makefile`
2. Load the executable file using the Visual C++ IDE command:

   File > Open Workspace

Viewing Object Reference Variables

While using the Visual C++ IDE debugger, you can view the value of an object from an object-reference variable. To do this, you use the debugger to get the object's database address, represented by its object identifier (OID), and then, after the transaction is complete, use `oodump` or `oodebug` to view the object's contents.

EXAMPLE Assume your application declares an object-reference variable `oopvar` to a persistent object. To view this variable:

1. Select `oopvar` and then use the Visual C++ QuickWatch dialog to display the following:

```
-oopvar = {...}
   -ooObj_ooRef = {...}
      -ooIdBase = {...}
         _DB = 3
         _OC = 2
         _page = 2
         _slot = 61
```

   This represents the OID `3-2-2-61`, the federated database address of the persistent object pointed to by `oopvar`.

2. When the transaction has completed, view the object contents using `oodump`:

```
C:\> oodump -id 3-2-2-61 -r one infoNet
```

Viewing Handle Variables

While using the Visual C++ debugger, you can view the contents of a persistent object from a handle that references it. To do this, you use the `ooprint` convenience function. See Chapter 5, "Debugging a Federated Database," in the Objectivity / DB administration book.

Sample Applications

Objectivity / C++ provides a number of sample applications in subdirectories of the *`installDir`*`\samples` directory. Most of these applications can be built either using `nmake` from the command line, or using the project file (`.mak`) provided in each subdirectory. Inspect the `makefile` and `makefile.mvc` files in each subdirectory for the information (such as the federated database file and boot file names) that is expected by the sample applications.

Table A-17 briefly describes the sample applications in *installDir*\samples.

Table A-17: Sample Objectivity/C++ Applications

Subdirectory of *installDir*\samples	Contains Application Demonstrating
abstr_mi	Persistent abstract class and multiple inheritance
cppdll	Dynamic link of the C++ interface
dll	Persistent DLL, exporting a single function or all persistent data and functions
hndasc	Persistent C++ implementation of the Library Object Model using handles and associations
mfclom	Persistent C++ implementation of the Library Object Model using the MFC library

B

Uninstalling Objectivity Products

You can uninstall Objectivity products—for example, when preparing to install a new release.

NOTE If you are uninstalling Objectivity / DB or Objectivity / SQL++ on Windows NT or Windows 2000, you must log on as administrator or as a user with equivalent privileges. These privileges are required for modifying network services.

To uninstall an Objectivity product:

1. In Control Panel, double-click **Add/Remove Programs**.
2. Select the Objectivity product and version you want to uninstall.
3. Click **Add/Remove**.

Uninstalling an Objectivity product removes all product files and empty product directories. If the product includes servers, these are unregistered as network services on Windows NT and Windows 2000.

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 a 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

E

F

H

I

L

M

N

O

P

R

S

T

U

V

W