



Installation and Platform Notes for Windows

Release 10.2

Installation and Platform Notes for Windows

Part Number: 10.2-IWIN-0

Release 10.2, October 13, 2011

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

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

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

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

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

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

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

Contents

About This Book	9
Audience	9
Organization	9
Conventions and Abbreviations	10
Getting Help	11
Chapter 1 Objectivity/DB Installation	13
System Requirements	14
Accessing Objectivity Installation Files	15
Installing Objectivity/DB	15
Preparing to Use Objectivity/DB	17
Setting Up a License File	21
Starting Objectivity/Assist	22
Checking and Setting Up Objectivity Servers	24
Verifying and Configuring TCP/IP	26
Chapter 2 Objectivity/C++ Installation	29
System Requirements	29
Installing Objectivity/C++ or Objectivity/DDL	30
Preparing to Use Objectivity/C++ or Objectivity/DDL	30
Setting Up the Visual C++ IDE	31
Testing Objectivity/C++ Setup	34

Chapter 3	Objectivity/DB Active Schema for C++ Installation	37
	System Requirements	37
	Installing Active Schema for C++	38
	Preparing to Use Active Schema for C++	38
Chapter 4	Objectivity/.NET for C# Installation	41
	System Requirements	41
	Installing Objectivity/.NET for C#	42
	Preparing to Use Objectivity/.NET for C#	42
	Testing Objectivity/.NET for C# Setup	43
Chapter 5	Objectivity for Java Installation	45
	System Requirements	45
	Installing Objectivity for Java	46
	Preparing to Use Objectivity for Java	46
	Testing Objectivity for Java Setup	48
Chapter 6	Objectivity/DB Active Schema for Java Installation	51
	System Requirements	51
	Installing Active Schema for Java	52
	Preparing to Use Active Schema for Java	52
Chapter 7	Objectivity/Smalltalk for VisualWorks Installation	53
	System Requirements	53
	Installing Objectivity/Smalltalk for VisualWorks	54
	Preparing to Use Objectivity/Smalltalk for VisualWorks	54
	Setting Up VisualWorks	56
	Setting Up VisualWorks With ENVY/Developer	56
	Testing Objectivity/Smalltalk for VisualWorks Setup	57

Chapter 8	Objectivity/Python Installation	59
	System Requirements	59
	Installing Objectivity/Python	60
	Preparing to Use Objectivity/Python	60
	Testing Objectivity/Python Setup	61
Chapter 9	Objectivity/SQL++ Installation	63
	System Requirements	63
	Installing Objectivity/SQL++	64
	Preparing to Use Objectivity/SQL++	64
	Checking and Setting Up the ODBC Server	66
	Testing Interactive SQL++	68
	Preparing the ODBC Server for Testing	69
Chapter 10	Objectivity/SQL++ ODBC Driver Installation	71
	System Requirements	71
	Installing Objectivity/ODBC	72
	Preparing to Use Objectivity/ODBC	73
	Adding Objectivity/DB Data Sources	73
	Configuring TCP/IP	75
	Testing Objectivity/ODBC	76
Chapter 11	Objectivity/DB High Availability Installation	79
	System Requirements	79
	Installing Objectivity/HA	80
	Preparing to Use Objectivity/HA	80

Chapter 12	Objectivity/DB Parallel Query Engine Installation	83
	System Requirements	83
	Installing Objectivity/PQE	83
	Preparing to Use Objectivity/PQE	84
	Checking and Setting Up the Query Server	85
	Using Objectivity/PQE	86
Chapter 13	Objectivity/IPLS Installation	87
	System Requirements	87
	Installing Objectivity/IPLS	88
	Preparing to Use Objectivity/IPLS	88
	Using Objectivity/IPLS	89
Appendix A	C++ Application Development	91
	Linking Applications to Objectivity/DB	91
	Libraries for Dynamic Linking	92
	Automatic Linking of Objectivity/DB Libraries	92
	Linking Explicitly	93
	Compatibility With Other Runtime Libraries	94
	Linking to Additional Objectivity Products and Features	95
	Linking to Persistent Collections	95
	Linking to Objectivity/DB Active Schema for C++	96
	Automatic Loading of Objectivity/PQE	97
	Automatic Loading of Predicate Query Language Feature	97
	Linking to Object Qualification Library When Using ooObjectQualifiers	98
	Automatic Loading of Objectivity/IPLS	98
	Linking a Lock-Server Performance-Monitoring Program	99
	User-Created DLLs and Objectivity/DB	100
	Exporting Persistent Data From a User-Created DLL	100
	Linking User-Created DLLs to Objectivity/DB	101
	Incorporating a User-Created DLL	101

Application Programming Issues	102
Using the Microsoft Foundation Classes	102
Signal Handling	102
Handling Microsoft Visual C++ Name Decoration	102
Visual C++ Development	103
Preparing to Create an Objectivity/C++ Project	103
Creating a New Objectivity/C++ Project	104
Adding a Pre-Build Event to Create a Federation	106
Defining a Custom Build Rule to Process DDL Files	107
Adding Generated Files	109
Setting the Includes Search Path	110
Setting Link Properties	110
Building and Executing the Application	111
Debugging an Application	112
Preparing to Debug an Application	112
Using Memory-Checking Software	113
Appendix B Java Application Development	115
Java Command Line Options	115
64-bit Architectures	115
Appendix C Sample Applications	117
Overview of Sample Applications	118
Sample Federated Databases	119
Appendix D Uninstalling Objectivity Products	121
Appendix E Troubleshooting an Application	123
Index	125

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

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

Appendix D describes how to uninstall Objectivity products.

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

Conventions and Abbreviations

Navigation

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

Typographical Conventions

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

Abbreviations

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

Command Syntax Symbols

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

Command and Code Conventions

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

Getting Help

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

Technical Support Web Site

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

How to Reach Objectivity Customer Support

You can contact Objectivity Customer Support by:

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

Before You Call

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

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

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

Developer Network Web Site

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

<http://devnet.objectivity.com>

Objectivity/DB Installation

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

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

System Requirements

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

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

Hardware	Operating System ^a	Abbreviation	Addressing Mode
x86	Windows XP Professional	Windows XP	32-bit
	Windows Server 2003	Windows 2003	32-bit
	Windows Server 2008	Windows 2008	32-bit
	Windows 7	Windows 7	32-bit
x86_64	Windows XP Professional x64	Windows XP x64	64-bit
	Windows Server 2008 R2	Windows 2008	64-bit
	Windows 7 x64	Windows 7 x64	64-bit

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

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 26).
- One of the following versions of .NET Framework CLR:
 - .NET Framework 4.0 in Visual Studio 2010 on Windows XP and Windows 7.
 - .NET Framework 3.5 in Visual Studio 2008 with Service Pack 1 (SP1) on Windows XP and Windows 7.
 - .NET Framework CLR 2.0 in Visual Studio 2005 on Windows XP.
- Java runtime environment (required for running Objectivity/Assist, a graphical interface for browsing and editing for federated databases)

NOTE You must be able to log on as an administrator or equivalent user that has privileges to start and configure services.

Accessing Objectivity Installation Files

You can access the Objectivity installation files by downloading them from the Objectivity Technical Support Web site:

1. Access the download page:
`http://support.objy.com/new/download`
2. Click the link corresponding to your Windows platform and Microsoft Visual C++ version to download the correct installer.
3. Save the the distribution file (a .zip file) to your disk.
4. Uncompress the distribution file and extract the Objectivity installation files from it.

Installing Objectivity/DB

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

Before You Install

1. Verify that TCP/IP is installed, running, and configured properly; see “Verifying and Configuring TCP/IP” on page 26.
2. If you are installing multiple Objectivity products at this time, verify that the system requirements for those products are met; see the installation chapter for each such product in this book.
3. If you have not done so already, obtain the Objectivity installation files as described in “Accessing Objectivity Installation Files” above.
4. (Recommended) Uninstall any previous releases of Objectivity products currently on your computer; see Appendix D, “Uninstalling Objectivity Products.”

Note: If you do not manually uninstall the previous release, you may be presented with an upgrade prompt when you start the setup program. If you select the upgrade prompt, the new release is overlaid on your existing installation hierarchy.

Running the Objectivity Setup Program

NOTE You must log on as administrator or as a user with equivalent privileges.

1. Change to the directory containing the uncompressed installation files and double-click on `setup.exe` to start the setup program.
If you have an Objectivity distribution CD, place it in your computer's CD-ROM drive to start the setup program automatically. If the program fails to start, navigate to your CD-ROM drive and double-click on `setup.exe`
2. In Setup Type, streamline the selection of Objectivity products by choosing one of the following setup types:
 - Choose **Simple** to request a selection list of just the essential Objectivity products for C++, Java, .NET, Python, and Smalltalk application development.
 - Choose **Full** to install all Objectivity products, without further selection.
 - Choose **Custom** to request a selection list of all installable Objectivity products, from which you can install a custom subset.

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

3. In Setup Type, select the destination folder (*installDir*) in which to install your Objectivity products. Click **Next**.
4. In Select Features (**Simple** or **Custom** setup), select **Objectivity/DB** and any other listed products you want to install at this time. If you select a product that requires other products, the required products are selected automatically.
5. In Select Features (**Simple** or **Custom** setup), optionally expand **Objectivity/DB** and clear any Objectivity/DB features you do not want.
Clear **Services** (or a particular service) if you do not want Objectivity servers to be installed and started as local services—for example, because your plan for configuring Objectivity/DB does not call for running Objectivity servers on your computer.
6. Click **Next** and follow the prompts to complete the installation.
7. Restart your computer if prompted to do so.

Preparing to Use Objectivity/DB

After the setup program completes, perform the following steps:

1. Familiarize yourself with your installation by reading “What the Setup Program Does” on page 18.
2. Set up your Objectivity license file, if you have not already done so; see “Setting Up a License File” on page 21.
3. Verify that you can start Objectivity/Assist and perform any necessary setup; see “Starting Objectivity/Assist” on page 22.
4. Check the status of Objectivity servers and perform any necessary configuration; see “Checking and Setting Up Objectivity Servers” on page 24.
5. If you installed multiple Objectivity products, read the installation chapter for each product, and follow the steps in the chapter’s “Preparing to Use” section.
6. Familiarize yourself with Objectivity online books. To do so, click **Start > Programs (or All Programs) > Objectivity x.x > Documentation > Objectivity Books**. A PDF file is displayed containing links to the online books.

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

7. Read:
 - *Objectivity Release Notes* for new and changed features. Click **Start > Programs (or All Programs) > Objectivity x.x > About This Release**.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.
8. Use Objectivity/Assist to perform the Objectivity/DB Basics Tutorial for getting started with Objectivity/DB concepts.

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

What the Setup Program Does

For Objectivity/DB, the setup program:

- Creates an Objectivity submenu under **Programs** (or **All Programs**) in the **Start** menu, and adds the program shortcuts shown in Table 1-2.

Table 1-2: Program Shortcuts Installed by Objectivity Setup

Shortcut		Description
About This Release		Displays <i>Objectivity Release Notes</i> (in PDF), which describes new and changed Objectivity products and documentation.
Objectivity Assist		Starts a tool for getting started with Objectivity/DB.
Documentation		
	Objectivity Books	Displays a starting point for viewing the online books (in PDF) for Objectivity products.
	Objectivity for java Books	Displays a starting point for viewing the online books (in HTML) for Objectivity for Java products.
	Objectivity Administration	Displays a starting point for viewing <i>Objectivity/DB Administration</i> (in HTML).
	Python Books	Displays a starting point for viewing the online books (in HTML) for Objectivity/Python products.
	ObjyNETcsharp.chm	Displays the Objectivity/.NET for C# help file.
Objectivity Network Services		Starts a tool for managing Objectivity servers.
Objectivity Developer Network		Opens the Objectivity Developer Network Web site, which provides technical information and resources.

Note: A submenu for ooBrowse is also added on 32-bit platforms, but this product is deprecated.

- Installs files in subdirectories of the Objectivity/DB installation directory *installDir*, as shown in Table 1-3.

Table 1-3: Objectivity/DB Release Files in *installDir*

Subdirectory	Contains
bin	Executables for Objectivity/DB tools and servers
	Dynamic link library (DLL) for Objectivity/DB tools and the Objectivity/DB kernel; DLL for object qualification; DLL for Objectivity/DB performance analyzer
	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) Subdirectories for HTML documentation sets Objectivity/.NET for C# help file
assist	Subdirectories for Objectivity/Assist plug-ins and features
	Executable for Objectivity/Assist
etc	Files and programs supporting various Objectivity tools and products.
include	Include file for the lock-server performance-monitoring interface
lib	Import library for custom lock-server performance-monitoring programs
plugins	Subdirectory for plugin specification files
samples	Sample database applications for demonstration and testing

- Modifies the system registry to set environment variables shown in Table 1-4.

Table 1-4: Basic Environment Variables Modified by Setup Program

Variable	Value
include	Search path for include files; used by Microsoft Visual C++. Set to contain <i>installDir\include</i> , which is created by installing Objectivity/C++. This component must precede the Visual C++ include directory in the search path.
lib	Search path for library files; used by Microsoft Visual C++. Modified to contain <i>installDir\lib</i> , which is created by installing 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 as well as Objectivity/Assist. Modified to contain <i>installDir\bin</i> and <i>installDir\assist</i> .

Unless you explicitly requested otherwise, the setup program also installs and starts Objectivity servers as services.

Setting Up a License File

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

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

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

1. Create a *default license file*:
 - a. Check whether a file called `oolicense.txt` already exists in your Objectivity/DB installation directory `installDir`. If so, move or rename that file.
 - b. Save your new Objectivity license as a text file called `oolicense.txt` in your Objectivity/DB installation directory `installDir`.

Note: If you received your encrypted license by electronic mail, you can simply save the attached file from your mail reader, or you can create an empty text file and paste the license text into it.
2. Make a backup copy of the new license file in another directory for safekeeping. A useful convention is to include the license’s identifier in the filename. (The identifier is included in the electronic mail message or other file accompanying the encrypted license.)

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

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

Starting Objectivity/Assist

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

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

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

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

Starting Objectivity/Assist

To start Assist:

1. Click **Start > Programs** (or **All Programs**) > **Objectivity x.x > Objectivity Assist**.
2. At the prompt, specify a *workspace directory* for Assist project files.

Assist Workspace Directory

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

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

Configuring Your Own Eclipse Platform

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

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

`installDir` Objectivity/DB installation directory

`eclipseInstallDir` Directory containing the Eclipse installation to be used—for example, `C:\eclipse`

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

Getting Started with Assist's Tutorials

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

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

Checking and Setting Up Objectivity Servers

The following Objectivity servers are provided with Objectivity/DB:

- The *lock server*, which manages concurrent access to persistent objects in one or more federated databases. For a complete description, see Chapter 8, “Using a Lock Server,” in the *Objectivity/DB Administration*.
- The Advanced Multithreaded Server (AMS), which allows remote database applications to access to local data files and journal files in a distributed Objectivity/DB system. For a complete description, see Chapter 9, “Advanced Multithreaded Server,” in *Objectivity/DB Administration*.

After the Objectivity setup program completes, you should check the status of the Objectivity servers on your computer. In some cases, you may need to perform additional setup steps, as described in the following subsections.

You can manage Objectivity servers using the Objectivity Network Services tool, which is described in Appendix A, “Running Objectivity Servers on Windows,” in *Objectivity/DB Administration*.

NOTE You must log in as administrator or as a user with equivalent privileges to make any changes through the Objectivity Network Services tool.

Checking Objectivity Server Status

To check the status of the Objectivity servers on your computer:

- Click **Start > Programs (or All Programs) > Objectivity x.x > Objectivity Network Services**.

The status of the Objectivity server is listed next to the server’s name.

If an Objectivity Server is Running

After a default Objectivity/DB installation, each server should be listed as **installed** and **running**. This means that the setup program successfully installed and started Objectivity servers as local services. When installed as services, these servers start automatically whenever the computer boots, and they run even when no user is logged in.

You can let each Objectivity server continue running, or you can stop the server using the Objectivity Network Services tool. However, you must restart a stopped server before you can run any database application that requires it.

In general, both Objectivity servers normally remain installed and running on any computer that supports Objectivity/DB development. As you refine the design of your distributed Objectivity/DB system, you will:

- Choose one or more computers on your network to be lock-server hosts for your federated databases.
- Decide whether or not to use AMS as your data-server software on the computers that host data files or journal files.

For a discussion of these decisions, see *Objectivity/DB Administration*. If you decide that you do not need to run an Objectivity server on a particular computer, you can use the Objectivity Network Services tool on that computer to uninstall the server as a service.

If an Objectivity Server is Stopped

If an Objectivity server is listed as **installed** and **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 the chapter for that server in *Objectivity/DB Administration*.

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 an Objectivity Server is Uninstalled

In certain cases, an Objectivity server might listed as **uninstalled**. This occurs if you deselected an Objectivity server during Objectivity/DB installation, or if the installation process was unable to install the server for some reason.

In this case, the setup program simply copies the executable to your computer but does not install the server as a service. If you later decide the server should run on this computer, you can use the Objectivity Network Services tool to install and start the service at any time.

Choosing a Logon Account for Objectivity Servers

The lock server and AMS log on to the local system account by default. For security purposes, you should consider configuring each Objectivity server to use a special-purpose account that can be granted the minimum necessary permissions. Note that when an application uses AMS, the logon account for AMS establishes the ownership of any data files created by the application.

For information about starting an Objectivity server with the required permissions, see the chapter for that server in *Objectivity/DB Administration*.

If You Use Windows Firewall

The Objectivity servers on your computer can serve an Objectivity/DB database application running anywhere on the same network. If an Objectivity server on your computer is to serve a remote application, and if your computer has Windows Firewall turned on, you must configure the Windows Firewall settings to enable traffic to the Objectivity server. See Appendix A, "Running Objectivity Servers on Windows," in *Objectivity/DB Administration*.

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).

Microsoft TCP/IP is included with all the Windows architectures in Table 1-1, 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 verify and configure Microsoft TCP/IP:

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, run the `ipconfig` command at a 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 Windows online help (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, `%systemroot%\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.

Accommodating Remote Data Access Through NFS

You can use your Windows computer to run Objectivity/DB applications that access data on remote UNIX workstations. If you plan to use Network File System (NFS) as the data-server software on any of these workstations, you may need to adjust the data packet size on your Windows computer.

In particular, you should check whether the TCP/IP protocol stack on your Windows computer requires a smaller data packet size than 8192 bytes, which is the default used by Objectivity/DB with NFS. Furthermore, in a congested network, a Remote Procedure Call (RPC) timeout error message may also indicate that the data packet size is too large.

To adjust the data packet size:

- Set the environment variable `OO_NFS_MAX_DATA`.

NOTE You do not need to adjust the data packet size if you use AMS instead of NFS. In general, AMS is the recommended data-server software on any host because of its performance, flexibility, and ease of use. See Chapter 9, “Advanced Multithreaded Server,” in *Objectivity/DB Administration*.

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 a federated database.
- Objectivity/DDL is a preprocessor for converting Data Definition Language (DDL) files into a schema of data types in a federated database. The DDL processor also produces source and header files for these data types.

System Requirements

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

Software Requirements

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

- Objectivity/DB (see Chapter 1)
- One of the following versions of Microsoft Visual C++:
 - Microsoft Visual C++ 8.0
 - Microsoft Visual C++ 9.0
 - Microsoft Visual C++ 10.0

See the release information 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++.

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 29.
2. Start the setup program (`setup.exe`) for installing Objectivity products.
If you have an Objectivity distribution CD, place it in your computer’s CD-ROM drive to start the setup program automatically. If the program fails to start, navigate to your CD-ROM drive and double-click on `setup.exe`
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. In Select Features, expand **Objectivity/C++** and select **Objectivity/C++ base**. 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 be licensed for every product you install.

5. Click **Next** and follow the prompts to complete the installation.

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

After the setup program completes, perform the following steps:

1. Familiarize yourself with your installation by reading “What the Setup Program Does” on page 31.
2. Set up the Visual C++ Integrated Development Environment (IDE) for use with Objectivity/C++; see “Setting Up the Visual C++ IDE” on page 31.
3. Test your installation by running the provided C++ sample application; see “Testing Objectivity/C++ Setup” on page 34.
4. (Optional) Use Objectivity/Assist to perform the C++ Tutorial for getting started with Objectivity/C++ application development.
5. 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 > Programs (or All Programs) > Objectivity x.x > About This Release**.

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

What the Setup Program 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 ^a
doc	PDF files for online books: <i>Objectivity/C++ Data Definition Language</i> , <i>Objectivity/C++ Programmer's Guide</i> , and <i>Objectivity/C++ Programmer's Reference</i>
include	Include files for the C++ programming interface
lib	Import libraries for C++ database applications ^a
samples\cxx	Sample C++ database applications for demonstration and testing

a. See "Linking Applications to Objectivity/DB" on page 91.

- Verifies the settings of the `path`, `lib`, and `include` environment variables. If necessary, these variables should be set to the values shown in Table 1-4.

Setting Up the Visual C++ IDE

After installation completes, you should set up the Visual C++ IDE to work with Objectivity/C++ and Objectivity/DDL. To do so, you must set the search path to find various Objectivity/C++ files. When you have done this, you can:

- Run the Objectivity/C++ sample applications; see "Testing Objectivity/C++ Setup" on page 34.
- Set up your own projects by following the steps in "Visual C++ Development" on page 103.

Setting the Visual C++ 8.0 or 9.0 IDE Search Paths

1. Log on to your user account.
2. click **Tools > Options**, expand **Projects and Solutions**, and then select the **VC++ Directories** property page.
3. In the Show Directories For list, click **Include files** and enter the following path in the Directories list:
`installDir\include`
4. In the Show Directories For list, click **Executable files** and enter the following path in the Directories list:
`installDir\bin`
5. In the Show Directories For list, click **Library files** and enter the following path in the Directories list:
`installDir\lib`

Setting the Visual C++ 10.0 IDE Search Paths

Visual C++ 10.0 lets you set search paths on a per project basis, or globally for all projects for a given user on a given machine.

To set search paths for a particular project:

1. Log on to your user account.
2. Open the project for which you want to set the search paths.
3. In the Solution Explorer, right click on the project name and choose **Properties** to access the Property Pages dialog.
4. Choose the configuration for which you want to set the search paths, for example, **All Configurations** and **Win32**.
5. Click the **VC++ Directories** Configuration Property.
6. Set the Search paths:
 - Click **Executable Directories**, then click the downward arrow to the right of the text-entry field, choose **Edit** from the drop-down menu, then click the folder icon and add the following path:
`installDir\bin`
 - Click **Include Directories**, then click the downward arrow to the right of the text-entry field, choose **Edit** from the drop-down menu, then click the folder icon and add the following path:
`installDir\include`
 - Click **Library Directories**, then click the downward arrow to the right of the text-entry field, choose **Edit** from the drop-down menu, then click the folder icon and add the following path:
`installDir\lib`

To set search paths globally:

1. Log on to your user account.
2. Open any VC++ 10 project.
The settings will be global—it doesn't matter which project you open.
3. From Visual Studio, choose **View > Other Window > Property Manager**.
Note: The **Property Manager** menu might be directly on the **View** menu, depending on your Visual Studio profile.
4. In the Property Manager pane, expand the type of build for which you want to set global search paths, for example, **Release | Win32**.
5. Double click on **Microsoft.Cpp.Win32.user** to access a property page for setting global search paths.
6. Click **VC++ Directories** under Common Properties.
7. Specify the search paths as described in step 6 in the previous procedure.

Testing Objectivity/C++ Setup

If you installed both Objectivity/C++ and Objectivity/DDL, you can test whether they are set up correctly by using your Visual C++ IDE to build and run the HelloWorld sample application provided with the installation.

When you build the sample application, the provided Visual C++ project creates an Objectivity/DB federated database. Executing the sample application populates the federated database with persistent objects.

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

Building and Executing the Sample Application

1. If you have not already done so, set up a default license file for your current Objectivity license; see “Setting Up a License File” on page 21.
2. Make sure the lock server is running on your computer; see “Checking and Setting Up Objectivity Servers” on page 24.
3. Set up the Visual Studio IDE to work with Objectivity/C++; see “Setting Up the Visual C++ IDE” on page 31.
4. Click **File > Open Project/Solution** and specify the `helloWorld` solution in the *installDir*\samples\cxx\helloWorld directory.
5. Click **Build > Build helloWorld**.
6. Click **Debug > Start Without Debugging** and see “Demo Results” below.

Demo Results

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

```
Welcome to the Objectivity HelloWorld Example
No Objectivity FD Bootfile passed as argument; using default:
../..data/helloWorld/HelloWorld.boot
Create database
Create HelloObject and set message
Looking for object with name HelloWorld
Found it!: HelloWorld
The HelloWorld test has PASSED.
```

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

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

If the installation test failed:

1. Verify your environment-variable settings, Visual C++ IDE search path, and default Objectivity license file.
2. Correct any errors, click **Build > Clean** or **Build > Clean helloWorld**, and then build the application again.

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

Objectivity/DB Active Schema for C++ Installation

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

System Requirements

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

Software Requirements

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

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

Installing Active Schema for C++

To install Active Schema for C++:

1. Verify that all required software has been completely and correctly installed; see “Software Requirements” on page 37.
2. Start the setup program (`setup.exe`) for installing Objectivity products.
If you have an Objectivity distribution CD, place it in your computer’s CD-ROM drive to start the setup program automatically. If the program fails to start, navigate to your CD-ROM drive and double-click on `setup.exe`
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 **Active Schema for C++**. If a required product is not yet installed, it is selected and installed automatically.

NOTE You must be licensed for every product you install.

5. Click **Next** and follow the prompts to complete the installation.

Preparing to Use Active Schema for C++

After the setup program completes, perform the following steps:

1. Familiarize yourself with your installation by reading “What the Setup Program Does” on page 39.
2. Read:
 - *Objectivity Release Notes* for new and changed features. Click **Start > Programs (or All Programs) > Objectivity x.x > About This Release**.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

What the Setup Program Does

For Active Schema for C++, the setup program installs files in subdirectories of the Objectivity/DB installation directory *installDir*, as shown in Table 3-1.

Table 3-1: Active Schema Release Files in *installDir*

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

a. "Linking to Objectivity/DB Active Schema for C++" on page 96.

Objectivity/.NET for C# Installation

This chapter describes the requirements and steps for installing Objectivity/.NET for C# on a Windows platform. Objectivity/.NET enables you to use .NET programming languages to write applications that store and manipulate persistent data in a federated database. The current release of this product supports the C# programming language.

System Requirements

You can install Objectivity/.NET for C# on any of the Windows architectures listed in Table 1-1 on page 14.

Software Requirements

Objectivity/.NET for C# requires that the following software be installed on your computer:

- Objectivity/DB (see Chapter 1)
- Microsoft Visual Studio 2008 with Service Pack 1 (SP1) or Microsoft Visual Studio 2010

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

Installing Objectivity/.NET for C#

To install Objectivity/.NET for C#:

1. Verify that all required software has been completely and correctly installed; see “Software Requirements” on page 41.
2. Start the setup program (`setup.exe`) for installing Objectivity products.
If you have an Objectivity distribution CD, place it in your computer’s CD-ROM drive to start the setup program automatically. If the program fails to start, navigate to your CD-ROM drive and double-click on `setup.exe`
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. In Select Features, select **Objectivity/.NET for C#**. If a required product is not yet installed, it is selected and installed automatically.

NOTE You must be licensed for every product you use.

5. Click **Next** and follow the prompts to complete the installation.

Preparing to Use Objectivity/.NET for C#

After the setup program completes, perform the following steps:

1. Familiarize yourself with your installation by reading “What the Setup Program Does” on page 43.
2. Test your installation by running the provided .NET sample application; see “Testing Objectivity/.NET for C# Setup” on page 43.
3. Read:
 - Chapter 2, “Developing a Sample Schema,” of *Objectivity/DB Schema Development*, which explains how to set up a Microsoft Visual Studio Project for .NET/C# and how to set up and the Objectivity/DB Persistence Designer. This chapter also explains how to create a schema design using the Persistence Designer.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

What the Setup Program Does

For Objectivity/.NET, the setup program:

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

Table 4-1: Objectivity/.NET Release Files in *installDir*

Subdirectory	Contains
bin	Assembly file (<i>Objectivity.DBxx.dll</i>) for building C# database applications. Assembly file (<i>Objectivity.DB.Linqxx.dll</i>) for LINQ queries in C# database applications. Assembly file (<i>ObjyPersistentDesignWizard.dll</i>) for the Objectivity/DB Persistence Designer. XML file (<i>Objectivity.DBxx.XML</i>) with summary of information about Objectivity types.
bin\debug	Debug assembly file (<i>Objectivity.DBxx.dll</i>) for building C# database applications.
doc	Compiled help file (.chm) for the <i>Objectivity/.NET for C# Programmer's Reference</i> . PDF file for <i>Objectivity/DB Schema Development</i> .
samples\NET-csharp	Sample C# database applications for demonstration and testing.

- Verifies the settings of the *path*, *lib*, and *include* environment variables. If necessary, these variables should be set to the values shown in Table 1-4.

Testing Objectivity/.NET for C# Setup

You can test whether Objectivity/.NET for C# is set up correctly by using Visual Studio 2008 or Visual Studio 2010 to build and run the HelloWorld sample application provided with the installation.

When you build the sample application, the provided Visual Studio solution creates an Objectivity/DB federated database. Executing the sample application populates the federated database with persistent objects.

You can inspect the various sample applications in the directory *installDir\samples\NET-csharp* to see how to use Objectivity/.NET for C# features; see "Overview of Sample Applications" on page 118.

Building and Executing the Sample Application

1. If you have not already done so, set up a default license file for your current Objectivity license; see “Setting Up a License File” on page 21.
2. Make sure the lock server is running on your computer; see “Checking and Setting Up Objectivity Servers” on page 24.
3. In Visual Studio, click **File > Open Project/Solution** and specify the `helloWorld` solution in the following directory:
`installDir\samples\NET-csharp\helloWorld.`
4. Set the `helloWorld` project to reference the required assemblies:
 - a. In the Solution Explorer, right-click the `helloWorld` project and click **Add Reference**.
 - b. In the Add Reference dialog, click the `.NET` tab. From the list, select `Objectivity.DB` and click **OK**.
5. Click **Build > Build Solution**.
6. Click **Debug > Start Without Debugging** and see “Demo Results” below.

Demo Results

If Objectivity/.NET for C# is set up correctly, the sample application displays messages such as:

```
Welcome to the Objectivity HelloWorld Example
Using default Objectivity FD specified in configuration file:
..\..\data\helloWorld.boot
Create Database
Create HelloObject and set message
Looking for object with name HelloWorld
Found it!: HelloWorld
The HelloWorld test has PASSED
```

If Objectivity/.NET for C# is not set up correctly, the sample application displays:

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

If the installation test failed:

1. Verify your environment-variable settings and default Objectivity license file.
2. Verify that you ran the sample application as recommended in step 6. Doing so establishes the correct current directory, enabling the boot file to be found using the expected relative path.
3. Correct any errors, click **Build > Clean** or **Build > Clean helloWorld**, and then build the application again.

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

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 federated database.

System Requirements

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

Software Requirements

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

- Objectivity/DB (see Chapter 1)
- The supported Java Development Kit (JDK) for your architecture.
See the release information on the Objectivity Technical Support Web site for the currently supported JDK version. Contact Objectivity Customer Support to get access to this Web site.
- A Web browser to view Objectivity for Java online books in HTML format.

The Java Connection Architecture (JCA) package of Objectivity for Java requires that the following software be installed:

- One of the following application servers:
 - BEA WebLogic Server 8.1
 - IBM WebSphere Application Server 6.0 or 6.1

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 45.
2. Start the setup program (`setup.exe`) for installing Objectivity products.
If you have an Objectivity distribution CD, place it in your computer’s CD-ROM drive to start the setup program automatically. If the program fails to start, navigate to your CD-ROM drive and double-click on `setup.exe`
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 be licensed for every product you install.

5. Click **Next** and follow the prompts to complete the installation.

Preparing to Use Objectivity for Java

After the setup program completes, perform the following steps:

1. Familiarize yourself with your installation by reading “What the Setup Program Does” on page 47.
2. (Optional) Test your installation by running the provided sample application; see “Testing Objectivity for Java Setup” on page 48.
3. (Optional) Use Objectivity / Assist to perform the Java Tutorial for getting started with Objectivity for Java application development.
4. If you are using Objectivity for Java with the JCA package, perform the steps in the package description for `com.objy.jca` in the online HTML book, *Objectivity for Java Programmer’s Reference*.
5. Read:
 - Appendix B, “Java Application Development,” in this book for platform-specific information about using Objectivity for Java.
 - *Objectivity Release Notes* for new and changed features. Click **Start > Programs (or All Programs) > Objectivity x.x > About This Release**.

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

What the Setup Program 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	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**.java java\samples***.java	Java database applications used as examples in the guide
lib	oojava.jar	Library executable
samples	java**.java	Java database applications for demonstration and testing
src	java\src.zip	Library source

- Adds the Objectivity for Java library path, *installDir\lib\oojava.jar*, to the CLASSPATH environment variable. (For some application development environments, you must specify CLASSPATH from within the tool.)
- Adds the **Objectivity for Java Books** shortcut to the **Objectivity x.x > Documentation** submenu. This shortcut displays an index page that provides access to the HTML online books through your Web browser.

Testing Objectivity for Java Setup

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

You can inspect the various sample applications in subdirectories of the directory `installDir\samples\java` to see how to use various Objectivity for Java features; see “Overview of Sample Applications” on page 118. (You can also inspect the applications in the directory `installDir\doc\java\samples`, which are used as part of the Objectivity for Java documentation.)

To build and run the sample application from a command prompt:

1. If you have not already done so, set up a default license file for your current Objectivity license; see “Setting Up a License File” on page 21.
2. Make sure the lock server is running on your computer; see “Checking and Setting Up Objectivity Servers” on page 24.
3. Change to the data directory for sample applications. At a command prompt, enter:

```
cd installDir\samples\data
```

4. Check whether the data directory already has a subdirectory called `helloWorld`. Create the subdirectory if necessary, and change to it.
cd `helloWorld`
5. Check whether the `helloWorld` directory already contains the HelloWorld federated database—for example, because you have already run the Objectivity/C++ sample application. If the files `HelloWorld.fdb` and `HelloWorld.boot` already exist, skip to step 7; otherwise, continue with step 6.

6. Create the HelloWorld federated database. At the command prompt, enter:

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

where *hostName* is the name of your computer.

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

7. Change to the directory containing the sample application source files. At a command prompt, enter:

```
cd installDir\samples\java\helloWorld\src
```

8. Build the sample application. At the command prompt, enter:

```
javac *.java
```

9. Run the sample application. At the command prompt, enter:

```
java Main ..\..\..\data\helloWorld\HelloWorld.boot
```

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

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

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

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

If the installation test failed:

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

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

Objectivity/DB Active Schema for Java Installation

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

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

System Requirements

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

Software Requirements

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

- Objectivity/DB (see Chapter 1)
- Objectivity for Java (see Chapter 5)

Installing Active Schema for Java

To install Active Schema for Java:

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

Preparing to Use Active Schema for Java

After the setup program completes, perform the following steps:

1. If you have not already done so, perform the steps for setting up Objectivity for Java; see “Preparing to Use Objectivity for Java” on page 46.
2. If your Objectivity license does not authorize you to use Active Schema for Java, contact your account manager to purchase appropriate licensing.
3. With an HTML browser, familiarize yourself with the product documentation; see Table 6-1.
4. Read:
 - *Objectivity Release Notes* for new and changed features. Click **Start > Programs** (or **All Programs**) > **Objectivity x.x** > **About This Release**.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

What the Setup Program Does

Active Schema for Java software is incorporated in the files of Objectivity for Java, so the setup program does not install separate Active Schema for Java files.

The online reference for Active Schema for Java is incorporated in the Objectivity for Java Programmer’s Reference, and is located within Objectivity/DB installation directory *installDir*, as shown in Table 6-1.

Table 6-1: Active Schema for Java Documentation in *installDir*

Directory	Files	Contains
doc	java\index.html	Document index
	java\api\com\objy\as\...*.html	Online reference

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 federated database.

System Requirements

You can install Objectivity/Smalltalk for VisualWorks on any of the Windows architectures listed in Table 1-1 on page 14, *except* Windows XP x64.

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

Installing Objectivity/Smalltalk for VisualWorks

To install Objectivity/Smalltalk for VisualWorks:

1. Verify that all required software has been completely and correctly installed; see “Software Requirements” on page 53.
2. Start the setup program (`setup.exe`) for installing Objectivity products.
If you have an Objectivity distribution CD, place it in your computer’s CD-ROM drive to start the setup program automatically. If the program fails to start, navigate to your CD-ROM drive and double-click on `setup.exe`
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 be licensed for every product you install.

5. Click **Next** and follow the prompts to complete the installation.

Preparing to Use Objectivity/Smalltalk for VisualWorks

After the setup program completes, perform the following steps:

1. Familiarize yourself with your installation by reading “What the Setup Program Does” on page 55.
2. Set up your image by performing the steps in one of the following sections:
 - “Setting Up VisualWorks” on page 56
 - “Setting Up VisualWorks With ENVY/Developer” on page 56
3. Test the installation of Objectivity/Smalltalk for VisualWorks; see “Testing Objectivity/Smalltalk for VisualWorks Setup” on page 57.

4. Read:

- *Objectivity Release Notes* for new and changed features. Click **Start > Programs** (or **All Programs**) > **Objectivity x.x > About This Release**.
- The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

What the Setup Program Does

For Objectivity/Smalltalk for VisualWorks, the setup program installs files in subdirectories of the Objectivity/DB installation directory *installDir*, as shown in Table 7-1.

Table 7-1: Objectivity/Smalltalk for VisualWorks Release Files in *installDir*

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

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. Parcel in the file `installDir\etc\smalltalk\objyDB.pcl`.
3. Save the image.
4. File in your development code.
5. (Optional) Delete the file `installDir\etc\smalltalk\objyDB.dat` to free disk space. This file is only used with ENVY/Developer.
6. (Optional) Test the installation; see “Testing Objectivity/Smalltalk for VisualWorks Setup” on page 57.

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\smalltalk\objyDB.dat`.

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

4. Use the option **load with required maps** for the configuration map **Objectivity/DB**.
5. Save the image.
6. File in your development code.
7. (Optional) Test the product installation on ENVY/Developer; see “Testing Objectivity/Smalltalk for VisualWorks Setup” on page 57.
8. (Optional) After importing the configuration maps and sample application code (see *Objectivity/Smalltalk for VisualWorks*) 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. To do so:

- In your VisualWorks image, evaluate the expression:

```
OoReleaseInstallUtility verifyInstall
```

This method sends output to the Transcript.

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

1. Set up a default license file for your current Objectivity license, if you have not already done so; see “Setting Up a License File” on page 21.
2. Create a federated database; see Chapter 4, “Federated Database Tasks,” of *Objectivity/DB Administration*.

3. In your VisualWorks image, evaluate the following expression:

```
OoReleaseInstallUtility verifyInstall: bootFilePath
```

where *bootFilePath* is the path to the boot file of a federated database.

Objectivity/Python Installation

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

System Requirements

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

Software Requirements

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

- Objectivity/DB (see Chapter 1)
 - The version of Python compatible with the current release of Objectivity/DB
- Note:** See the Objectivity Technical Support Web site for the compatible version of Python for the current release of Objectivity/DB. Contact Objectivity Customer Support to get access to this Web site.

Installing Objectivity/Python

To install Objectivity/Python:

1. Verify that all required software has been completely and correctly installed (see “Software Requirements” above).
2. Start the setup program (`setup.exe`) for installing Objectivity products.
If you have an Objectivity distribution CD, place it in your computer’s CD-ROM drive to start the setup program automatically. If the program fails to start, navigate to your CD-ROM drive and double-click on `setup.exe`
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/Python**. If a required product is not yet installed, it is selected and installed automatically.

NOTE You must be licensed for every product you install.

5. Click **Next** and follow the prompts to complete the installation.

Preparing to Use Objectivity/Python

After the setup program completes, perform the following steps:

1. Familiarize yourself with your installation by reading “What the Setup Program Does” on page 61.
2. If necessary, add the Objectivity/Python library path, `installDir\bin`, to the `PYTHONPATH` environment variable. This is necessary if your application development environment requires you to specify the `PYTHONPATH` environment variable from within that environment.
In most instances, manual setting of the `PYTHONPATH` is not required, as the setup program is configured to automatically set the `PYTHONPATH` environment variable for you.
3. (Optional) Test your installation by running the provided sample application; see “Testing Objectivity/Python Setup” on page 61.
4. (Optional) See “Getting Started with Objectivity/Python” to learn about Python application development.
5. Read:
 - *Objectivity Release Notes* for new and changed features. Click **Start > Programs (or All Programs) > Objectivity x.x > About This Release**.

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

What the Setup Program Does

The Objectivity/Python setup program:

- Installs files in subdirectories of the Objectivity/DB installation directory *installDir*, as shown in Table 8-1.

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

Subdirectory	Contains	
doc	python\index.html	Document index
	python\api*.html	<i>Objectivity/Python Programmer's Reference</i> (HTML)
	python\guide\pyGettingStarted.html	<i>Getting Started with Objectivity/Python</i>
bin	Dynamic linking library (DLL) for linking Objectivity/Python applications	
samples	python**.py	Sample Objectivity/Python database applications for demonstration and testing.

- Sets the PYTHONPATH environment variable.
- Adds the **Objectivity/Python Books** shortcut to the Objectivity submenu. This shortcut displays an index page that provides access to the HTML online books through your Web browser.

Testing Objectivity/Python Setup

You can test whether Objectivity/Python is set up correctly by executing the `rentalFleet` demo application provided with the installation. Executing the sample application creates and populates an Objectivity/DB federated database with persistent objects.

To build and run the sample application from a command prompt:

1. If you have not already done so, set up a default license file for your current Objectivity license; see “Setting Up a License File” on page 21.
2. Make sure the lock server is running on your computer; see “Checking and Setting Up Objectivity Servers” on page 24.

3. Change to the directory for sample applications. At a command prompt, enter:
`cd installDir\samples\python\rentalFleet`
4. Run the sample application in the rentalFleet directory. At the command prompt, enter:
`rf.py`

NOTE If the federated database already exists, the sample application will delete it and create a new one for you.

If Objectivity/Python is set up correctly, the sample application displays messages such as:

```
installDir\Samples\python\rentalFleet>rf.py
Begin: Rental Fleet sample for Python
creating logs directory
creating fd directory
Creating FD:
installDir\Samples\python\rentalFleet\fd\RentalFleet.boot
Connecting to:
installDir\Samples\python\rentalFleet\fd\RentalFleet.boot --
Succeeded
creating schema
schema created
populating
Data entry time = 1.81299996376
End: Rental Fleet sample for Python
```

If the installation test failed:

1. Verify your environment-variable settings and default Objectivity license file.
2. Correct any errors.
3. Rerun the Python script.

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

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 objects in an Objectivity/DB federated database, with extensions to accommodate the Objectivity/DB object model.

Objectivity/SQL++ has two components:

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

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

System Requirements

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

Software Requirements

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

- Objectivity/DB (see Chapter 1)

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

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

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

- Objectivity/SQL++ ODBC Driver (see Chapter 10)

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 63.
3. Start the setup program (`setup.exe`) for installing Objectivity products.
If you have an Objectivity distribution CD, place it in your computer’s CD-ROM drive to start the setup program automatically. If the program fails to start, navigate to your CD-ROM drive and double-click on `setup.exe`
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 be licensed for every product you install.

6. Click **Next** and follow the prompts to complete the installation.

Preparing to Use Objectivity/SQL++

After the setup program completes, perform the following steps:

1. Familiarize yourself with your installation by reading “What the Setup Program Does” on page 65.
2. Create an account with the username `syspe` and a password of your choice. The Objectivity/SQL++ database administrator will use this account. For more information about `syspe`, see *Objectivity/SQL++*.
3. Add a system environment variable called `OO_SQL_DIR` and set its value to the drive letter and pathname of the Objectivity/DB installation directory (`installDir`). Objectivity/SQL++ uses this environment variable to find help files and error-code files.
4. Check the status of the Objectivity/SQL++ ODBC server and perform any necessary configuration; see “Checking and Setting Up the ODBC Server” on page 66.
5. Test each Objectivity/SQL++ component you plan to use:
 - Test Interactive SQL++ by following the steps in “Testing Interactive SQL++” on page 68.

- 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 69.
6. Read:
- *Objectivity Release Notes* for new and changed features. Click **Start > Programs (or All Programs) > Objectivity x.x > About This Release.**
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

What the Setup Program Does

For Objectivity/SQL++, the setup program:

- Installs files in subdirectories of the Objectivity/DB installation directory *installDir*, as shown in Table 9-1.

Table 9-1: Objectivity/SQL++ Release Files in *installDir*

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

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

Checking and Setting Up the ODBC Server

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

After the Objectivity setup program completes, you should check the status of the ODBC server on your computer, and perform additional setup as necessary.

NOTE You manage the ODBC server using the Objectivity Network Services tool, which is described in Appendix A, “Running Objectivity Servers on Windows,” in *Objectivity/DB Administration*. You must log in as administrator or as a user with equivalent privileges to make any changes through the Objectivity Network Services tool.

Checking ODBC Server Status

To check the status of the ODBC server on your computer:

- ▶ Click **Start > Programs (or All Programs) > Objectivity x.x > Objectivity Network Services**.

The status of the ODBC server is listed next to the server’s name.

If the ODBC Server is Running

After installation, the ODBC server should be listed as **installed** and **running**. This means that the setup program has successfully installed and started the ODBC server as a local service. Consequently, the ODBC server will start automatically whenever the system boots, and will run even when no user is logged in.

You can let the ODBC server continue running, or you can stop it by clicking **Stop** in the Objectivity Network Services tool. To do this, you must be logged in as administrator or as a user with equivalent privileges.

A running ODBC server is required only when you are accessing Objectivity/DB federated 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 federated databases with Interactive SQL++.

If the ODBC Server is Stopped

If Objectivity Network Services lists the Objectivity/SQL++ ODBC server as **installed** and **stopped**, you should check whether another process is already

running on port 1990, which is the default port for the ODBC server. If another service already uses the default port, either reassign that service to a different port (recommended) or change the port for the ODBC server.

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

To confirm that a port conflict exists or to change the default port for the ODBC server, see Chapter 3, “Getting Started With ODBC,” in *Objectivity/SQL++*.

Choosing a User Account for the ODBC Server

At installation, the ODBC server is started under the local system account. For security purposes, you should consider running the ODBC server under a special-purpose user or group account that can be granted the minimum necessary permissions.

For information about starting an ODBC server with the required permissions, see Chapter 3, “Getting Started With ODBC,” in *Objectivity/SQL++*.

If You Use Windows Firewall

The ODBC server on your computer can serve ODBC-compliant applications running anywhere on the same network. If the ODBC server on your computer is to serve a remote application, and if your computer has Windows Firewall turned on, you must configure the Windows Firewall settings to enable traffic to the Objectivity server. See Appendix A, “Running Objectivity Servers on Windows,” in *Objectivity/DB Administration*.

(Optional) Changing the Log Directory

A running ODBC server creates temporary error-log files and cache files. By default, these files are written to the directory specified by the `TEMP` environment variable, if one is defined for your user account; otherwise to `C:\winnt`.

You can specify a nondefault log directory. To do so:

1. Log on as administrator or as a user with equivalent privileges.
2. Click **Start > Programs (or All Programs) > Objectivity x.x > Objectivity Network Services**.
3. Select the Objectivity/SQL++ ODBC server and click **Stop**.
4. When the ODBC has stopped, click **Configure**.
5. Enter or select the pathname of the log directory and click **OK**.
6. Restart the Objectivity/SQL++ ODBC server.

Testing Interactive SQL++

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

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

1. If you have not already done so, set up a default license file for your current Objectivity license; see “Setting Up a License File” on page 21.

2. Change to the Interactive SQL++ sample directory—for example, at a command prompt, enter:

```
cd installDir\samples\sql\ooisql
```

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

3. Edit `makefile` in the Interactive SQL++ sample directory:

- Set `INSTALL_DIR` to be the location of the Objectivity/DB installation directory—for example:

```
INSTALL_DIR = c:\objectivity
```

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

```
LS_HOST = myLockServerHost
```

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

5. Build the executables and the demo federated database. At a command prompt, enter:

```
nmake
```

6. Run the sample application. At the command prompt, enter:

```
demo
```

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

```

Creating the Objectivity/DB Database: .\create
Running OOISQL to create views:
    ooisql -input views.sql -user systpe -passwd dummy DEMO
Running OOISQL to test out various SQL statements:
    ooisql -input test.sql -user systpe -passwd dummy DEMO
Comparing the results.
Listing differences (if any). Null difference indicates -
    test passed.
Comparing files log and OOISQLOK
FC: no differences encountered

Test completed
Listing errors (if any). Null listing indicated - test passed
End of error list
Deleting ..\samples\sql\ooisqliog
Deleting ..\samples\sql\ooisqldiffs
Deleting ..\samples\sql\ooisqlerr

```

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

Preparing the ODBC Server for Testing

At some sites, a database administrator or a system administrator is responsible for installing Objectivity/SQL++, while individual users of ODBC-compliant client applications 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 2 through 6 beginning on page 68) to create and populate the demo federated database to be browsed. Be sure to leave the lock server running.
2. Check whether the lock server is running; start it, if necessary.

3. Check whether the 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 > Programs (or All Programs) > Objectivity x.x > Objectivity Network Services**.
 - c. Select the Objectivity/SQL++ ODBC server and click the **Start** button.
4. If the Objectivity/SQL++ ODBC Driver demo is to be performed by another user, give that user the TCP/IP name of the ODBC server host, the service name under which the ODBC server is running (`oosqlnw`), and the location and name of the boot file (`installDir\samples\sql\oosql\DEMO`).
5. Grant all access rights to all users for the tables in the demo federated database. If you omit this step, only the Objectivity/SQL++ database administrator (`systpe`) will have access to these tables. To grant access rights to all users:
 - a. Start Interactive SQL++ for the demo federated database:


```
oosql -user systpe -passwd adminpassword
      installDir\samples\sql\oosql\DEMO
```

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


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


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


```
commit work;
exit
```

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

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

Objectivity/SQL++ ODBC Driver Installation

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

See Chapter 9, “Objectivity/SQL++ Installation,” for information about the Objectivity/SQL++ ODBC server.

System Requirements

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

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

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

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

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

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. Start the setup program (`setup.exe`) for installing Objectivity products. If you have an Objectivity distribution CD, place it in your computer's CD-ROM drive to start the setup program automatically. If the program fails to start, navigate to your CD-ROM drive and double-click on `setup.exe`.
3. If any other Objectivity products are already installed on your computer, click **Modify**, click **Next**, and continue with step 4. Otherwise, if this is the first (or only) Objectivity product you are installing, select the directory (`installDir`) in which to install Objectivity/ODBC. Click **Next** and continue with step 4.
4. Select **Objectivity/SQL++ ODBC Driver**.

NOTE You must be licensed for every product you install.

5. Click **Next** and follow the prompts to complete the installation.

Preparing to Use Objectivity/ODBC

After the setup program completes, perform the following steps:

1. Familiarize yourself with your installation by reading “What the Setup Program Does” on page 73.
2. Add the desired Objectivity/DB federated databases as data sources. Follow the steps in “Adding Objectivity/DB Data Sources” on page 73.
3. Configure TCP/IP to enable Objectivity/ODBC to communicate with an Objectivity/SQL++ ODBC server. Follow the steps in “Configuring TCP/IP” on page 75.
4. (Optional) Test Objectivity/ODBC with an Objectivity/SQL++ ODBC server. Follow the steps in “Testing Objectivity/ODBC” on page 76.

What the Setup Program Does

The Objectivity/ODBC setup program:

- Installs files in subdirectories of the Objectivity/ODBC installation directory *installDir*, as shown in Table 10-1.

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

Subdirectory	Contains
bin	DLL for the ODBC driver
doc\odbcDriver.pdf	PDF file for the online book <i>Objectivity/SQL++ ODBC Driver User's Guide</i>
include	Include files for developing custom ODBC-compliant applications
samples\sql\odbc	Application for demonstration and testing

- 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* for it. In general, a data source identifies the data to be accessed and the means of accessing it (for example, host and network information). The data source you add for a federated database specifies its boot file and a host running a Objectivity/SQL++ ODBC server.

Before you add a data source for a federated database:

- Identify the Objectivity/SQL++ ODBC server that will access the federated database and obtain:
 - The TCP/IP name of the computer that runs the ODBC server.
 - The service name under which the ODBC server is running.
- Obtain the location and name of the federated database's boot file.
- Ensure that you have a valid user account on the host running the Objectivity/SQL++ ODBC server.

To add a data source for a federated database:

1. In Control Panel on the host where you installed Objectivity/ODBC (the driver), open Administrative Tools and click **Data Sources (ODBC)**.
(*Windows 7*) In Control Panel, open **System and Maintenance** (or **System and Security**), open **Administrative Tools**, and double-click **Data Sources (ODBC)**.
2. In ODBC Data Source Administrator, click **System DSN** (or **User DSN**, if you want to create a personal data source).
3. Click **Add**, select **Objectivity ODBC Driver** 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. The string may not exceed 32 characters.
Description	(Optional) A description of the data source.
Host	The TCP/IP name of the computer running the Objectivity/SQL++ ODBC server.
Database	The location and name of the boot file for the federated database, expressed as a fully qualified pathname beginning with a drive letter.
User ID	The username of your Objectivity/SQL++ account. This is normally your login account on the ODBC server host, provided that this account has been granted access rights to tables in the federated database (see your Objectivity/SQL++ database administrator). The Objectivity/SQL++ database administrator account (<code>sysupe</code>) has access to all tables.
Password	Password for the account you entered in the User ID field. The password you enter is <i>not</i> encoded before it is sent across the network.
Service	The service name of the ODBC server on its host (<code>oosqlnw</code>).

NOTE The **Host**, **Database**, and **Service** fields together must not exceed a string-length of 249 characters.

5. Click **OK**.
6. If you want to add other data sources, repeat steps 3 through 5.
7. When you are finished adding data sources, click **OK**.

Configuring TCP/IP

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

Identifying the ODBC Server's Host to TCP/IP

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

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

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

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 number 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. The Microsoft TCP/IP file location is:
`%systemroot%\system32\drivers\etc\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` is the TCP port number you found in step 1.

Testing Objectivity/ODBC

You can verify the correct operation of Objectivity/ODBC in combination with an Objectivity/SQL++ ODBC server. The following test (which requires a C++ development environment) compiles and links a sample ODBC application and uses that application to access a demo federated database. The demo federated database is provided with Objectivity/SQL++ on the server host.

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

To build and run the sample ODBC-compliant application:

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

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

2. On the Objectivity/ODBC (driver) host, add a data source for the demo federated database.
3. Go to the Objectivity/SQL++ ODBC samples directory—for example, in a command window, enter:
`cd installDir\samples\sql\odbc`
 You may wish to make a backup copy of this directory.
4. Check whether the lock server is running; start it, if necessary.
5. Build and run the sample application. At the command prompt, enter:
`nmake`

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

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


Objectivity/DB High Availability Installation

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

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

System Requirements

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

Software Requirements

Objectivity/HA 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 an autonomous partition or a replicated database; see “Checking and Setting Up Objectivity Servers” on page 24.

Installing Objectivity/HA

To install Objectivity/HA:

1. Verify that all required software has been completely and correctly installed; see “Software Requirements” above.
2. Start the setup program (`setup.exe`) for installing Objectivity products.
If you have an Objectivity distribution CD, place it in your computer’s CD-ROM drive to start the setup program automatically. If the program fails to start, navigate to your CD-ROM drive and double-click on `setup.exe`
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 High Availability**. If a required product is not yet installed, it is selected and installed automatically.

NOTE You must be licensed for every product you install.

5. Click **Next** and follow the prompts to complete the installation.

Preparing to Use Objectivity/HA

After the setup program completes, perform the following steps:

1. Familiarize yourself with your installation by reading “What the Setup Program Does” on page 81.
2. Verify that AMS is installed and running on every host that is to store an autonomous partition or a replicated database; see “Checking and Setting Up Objectivity Servers” on page 24. Although AMS need not be running when you create an original database image, you must start AMS before you can create additional database images.

3. Read:

- *Objectivity Release Notes* for new and changed features. Click **Start > Programs** (or **All Programs**) > **Objectivity x.x > About This Release**.
- The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

What the Setup Program Does

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

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

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

Objectivity/DB Parallel Query Engine Installation

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

System Requirements

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

Software Requirements

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

- Objectivity/DB (see Chapter 1)

Installing Objectivity/PQE

To install Objectivity/PQE:

1. Log 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 83.
3. Start the setup program (`setup.exe`) for installing Objectivity products.
If you have an Objectivity distribution CD, place it in your computer’s CD-ROM drive to start the setup program automatically. If the program fails to start, navigate to your CD-ROM drive and double-click on `setup.exe`

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/PQE**. If a required product is not yet installed, it is selected and installed automatically.

NOTE You must be licensed for every product you install.

6. Click **Next** and follow the prompts to complete the installation.

Preparing to Use Objectivity/PQE

After the setup program completes, perform the following steps:

1. Familiarize yourself with your installation by reading “What the Setup Program Does” on page 84.
2. Set up the query server by following the steps in “Checking and Setting Up the Query Server” on page 85.
3. Read:
 - *Objectivity Release Notes* for new and changed features. Click **Start > Programs (or All Programs) > Objectivity x.x > About This Release**.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

What the Setup Program Does

For Objectivity/PQE, the setup program:

- When you install Objectivity/PQE, its files are organized in subdirectories of the *installDir* directory, as shown in Table 12-1.

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

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

- Installs the query server as a service.

Checking and Setting Up the Query Server

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

You can run the query server on one or more hosts. You install query servers on additional hosts by installing Objectivity/PQE on each desired host. For security purposes, you should consider running the query server under a special-purpose user account in a group that can be granted just the minimum necessary permissions.

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

Checking the Query Server

To check the status of a query server on the current workstation:

- ▶ Click **Start > Programs (or All Programs) > Objectivity x.x > Objectivity Network Services**.

The status of a query server is listed next to the server's name.

To check the status of a query server on a particular workstation *hostname*:

- ▶ At a command prompt, enter the following command:
`ooqueryserver -check hostname`

If the Query Server is Running

After installation, the query server should be listed as **installed** and **running**. This means that the setup program has successfully installed and started the query server as a local server. Consequently, the query server will start automatically whenever the system boots, and will run even when no user is logged in.

You can let the query server continue running, or you can stop it by clicking **Stop** in the Objectivity Network Services tool. Alternatively, you can enter `ooqueryserver` with the `-stop` option at a command prompt.

If the Query Server is Stopped

If Objectivity Network Services lists the query server as installed and stopped, you should check whether another process is already running on port 6782, which is the default port of the query server. If another service already uses the default port, either reassign that service to a different port (recommended) or change the port for the query server. If you change the default port for the query

server, then you will have to make this change on *every* host that runs a process that interacts with this query server. For more information on changing a query server's port number, see Chapter 10, "Using a Query Server," in *Objectivity/DB Administration*.

Choosing a User Account for the Query Server

At installation, the query server is started under the local system account. For security purposes, you should consider running the query server under a special-purpose user or group account that can be granted the minimum necessary permissions.

If You Use Windows Firewall

The query server on your computer can serve query server compliant applications running anywhere on the same network. If the query server on your computer is to serve a remote application, and if your computer has Windows Firewall turned on, you must configure the Windows Firewall settings to enable traffic to the Objectivity server. See Appendix A, "Running Objectivity Servers on Windows," in *Objectivity/DB Administration*.

Using Objectivity/PQE

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

No extra steps are required to compile and link an Objectivity/PQE application. When your application initiates a parallel query, the appropriate Objectivity/PQE dynamic link library is loaded automatically at runtime. The Objectivity/PQE DLLs are listed in Table A-9 on page 97.

Objectivity/IPLS Installation

This chapter describes the requirements and steps for installing Objectivity/DB In-Process Lock Server (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 a separate lock-server process.

System Requirements

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

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 87.
2. Start the setup program (`setup.exe`) for installing Objectivity products.
If you have an Objectivity distribution CD, place it in your computer’s CD-ROM drive to start the setup program automatically. If the program fails to start, navigate to your CD-ROM drive and double-click on `setup.exe`
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**. If a required product is not yet installed, it is selected and installed automatically.

NOTE You must be licensed for every product you install.

5. Click **Next** and follow the prompts to complete the installation.

Preparing to Use Objectivity/IPLS

After the setup program completes, perform the following steps:

1. Familiarize yourself with your installation by reading “What the Setup Program Does” on page 89.
2. Read:
 - “Using Objectivity/IPLS” on page 89
 - *Objectivity Release Notes* for new and changed features. Click **Start > Programs (or All Programs) > Objectivity x.x > About This Release**.
 - The release information on the Objectivity Technical Support Web site for known problems and current configuration information. Contact Objectivity Customer Support to get access to this Web site.

What the Setup Program Does

For Objectivity/IPLS, the setup program installs files in subdirectories of the Objectivity/DB installation directory *installDir*, as shown in Table 13-1.

Table 13-1: 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 *Objectivity/C++ Programmer's Guide*, *Objectivity for Java Programmer's Guide*, or *Objectivity/Smalltalk for VisualWorks*.

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 99.

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

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/DDI books.

This appendix provides information about:

- Linking applications to [Objectivity/DB](#)
- Linking applications to [additional Objectivity products and features](#)
- [User-created DLLs](#) that export Objectivity/DB persistent data
- Objectivity/C++ application [programming issues](#)
- Developing Objectivity/C++ applications with [Visual C++ 8.0, Visual C++ 9.0, or Visual C++ 10.0](#)
- [Debugging](#) Objectivity/C++ applications
- [Memory-checking software](#)

See also Appendix C, “Sample 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 92.

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
<code>oodbi.lib</code>	Objectivity/DB import library
<code>oodbid.lib</code>	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
<code>oodbxx.dll</code>	Release version of the Objectivity/DB DLL (corresponds to the <code>oodbi.lib</code> import library)
<code>oodbxxd.dll</code>	Debug version of the Objectivity/DB DLL (corresponds to the <code>oodbid.lib</code> 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 (`oodbxx.dll`). In contrast, debug applications link to the debug-compatible import library (`oodbid.lib`) and use the corresponding DLL (`oodbxxd.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.

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. (`oo.h` is included in your application automatically when you run the DDL processor.)

Table A-3 shows the runtime libraries that are selected automatically when you link a release or debug version of an application.

Table A-3: 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

When 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 properties listed in Table A-4.

Table A-4: Link Properties Controlling Library Selection

Property	Value
Runtime Library	Multi-threaded DLL or Debug Multi-threaded DLL
Use of MFC	Use MFC in a Shared DLL

You should set one of these properties, but you do not need to set both; the MFC link property automatically sets the Visual C++ runtime link property. Regardless of the property you choose, you must set it to select a DLL (either for a release or a debug version).

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

To specify libraries explicitly in the Visual C++ IDE:

1. In the Solution Explorer, right-click your project and click **Properties**.
2. In the Property Pages dialog, click the **Linker** folder and click the **Input** property page.

3. Click the **Additional Dependencies** property and enter the desired libraries, preserving the link order shown above.
4. Click the **Ignore All Default Libraries** property and choose **Yes**.
5. In the Property Pages dialog, click **OK**.

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 Persistent Collections

For Release 9.3 (or later) Persistent Collections

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

For pre-Release 9.3 Persistent Collections

If any of the collections used by your application are implemented with the pre-Release 9.3 scalable collection structure (for example, `ooTreeSet`), you must include the header file `ooCollections.h`. Table A-5 lists the collection-specific import libraries; the release or debug import library is linked automatically when you include the `ooCollections.h` header file.

Table A-5: Dynamic Link Import Libraries For pre-Release 9.3 Persistent Collections

Library File	Description
<code>oooco.lib</code>	Multithreaded import library for Objectivity/C++ persistent collections
<code>ooocod.lib</code>	Debug version of the multithreaded import library for Objectivity/C++ persistent collections

Table A-6 lists the DLLs that must be available at runtime to applications that are linked dynamically to libraries in Table A-5.

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

DLL File ^a	Description
oocox.dll	DLL for Objectivity/C++ persistent collections
oocox.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/DB Active Schema for C++

Table A-7 lists the import libraries for Objectivity/DB Active Schema for C++ (Active Schema), 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 Active Schema header file.

Table A-7: Active Schema Dynamic Link Import Libraries

Library File	Description
ooas.lib	Active Schema multithreaded import library
ooas.d.lib	Debug version of the Active Schema multithreaded import library

Table A-8 lists the DLLs that must be available at runtime to applications that are linked dynamically to Active Schema.

Table A-8: Active Schema Dynamic Link Libraries

DLL File ^a	Description
ooasx.dll	Active Schema DLL
ooasx.d.dll	Debug version of the Active Schema DLL

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

Automatic Loading of Objectivity/PQE

When your application initiates a parallel query, the appropriate Objectivity/PQE dynamic link library is loaded automatically at runtime.

Table A-9 lists the DLLs that must be available at runtime to applications that use Objectivity/PQE.

Table A-9: Objectivity/PQE Dynamic Link Libraries

DLL File ^a	Description
oopqexx.dll	DLL for Objectivity/PQE

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

Automatic Loading of Predicate Query Language Feature

When your application uses the predicate-query language (PQL) to qualify persistent objects through a `scan` or `parallelScan` operation, or uses PQL to find qualified destination objects through a `linkName` method, the appropriate object qualification library is automatically loaded at runtime.

Table A-10 lists the DLLs that must be available at runtime to applications that use the predicate query language feature.

Table A-10: Predicate Query Language Dynamic Link Libraries

DLL File	Description
ooObjectQualification.dll ooObjectModel.dll ooSchemaModelInt.dll oopcre.dll ooasxx.dll	DLL for the predicate query language and the libraries on which it depends.

Linking to Object Qualification Library When Using ooObjectQualifiers

Table A-11 lists the import libraries for using an ooObjectQualifier to qualify individual objects.

The release or debug import library is linked automatically when you include the ooObjectQualifier.h header file.

Table A-11: Predicate Query Language Dynamic Link Import Libraries

Library File	Description
ooObjectQualification.lib	Predicate query language import library
ooObjectQualificationd.lib	Debug version of the predicate query language import library

Table A-12 lists the DLLs that must be available at runtime to applications that use ooObjectQualifier instances.

Table A-12: Predicate Query Language Dynamic Link Libraries

DLL File	Description
ooObjectQualification.dll ooObjectModel.dll ooSchemaModelInt.dll oopcre.dll oasxx.dll	DLL for the predicate query language and the libraries on which it depends.

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
ooiplsxx.dll	DLL for Objectivity/IPLS
ooiplsxxd.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
oolspmxx.dll	DLL for custom lock-server performance-monitoring programs
oolspmxxd.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.

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 `ooddtx.exe`, using the `-storage_specifier` option to pass the placeholder symbol to it. For example:

```
ooddtx -storage_specifier MY_STORAGE_SPECIFIER
```

When run with this option, `ooddtx.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 `ooddtx.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 101.

If you intend to use the `FreeLibrary` function, you should build DLLs with the following option: `OO_SKIP_AUTO_CLEANUP_AT_EXIT`. Using this option enables you to use the `FreeLibrary` function without the function invalidating your sessions during auto-cleanup. You can specify the option in your Visual C++ project definition. You must call `ooObjy :: shutdown` before exiting your application.

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++ collection classes instead). Furthermore, you should not include MFC header files in your DDL schema files.

Signal Handling

The Objectivity/DB predefined signal handler catches the following signals:

- SIGABRT; SIGFPE; SIGILL; SIGINT; SIGSEGV; SIGTERM

but does not catch the following signals and Windows events:

- SIGBREAK; the `ExitProcess` function; any unhandled exceptions that do not map directly to the caught signals; stack overflow

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 *Objectivity/DB Administration*.

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.

Visual C++ Development

This section describes the following tasks for creating new Objectivity/C++ projects using Visual C++ 8.0, Visual C++ 9.0, or Visual C++ 10.0.

- “Preparing to Create an Objectivity/C++ Project” on page 103
- “Creating a New Objectivity/C++ Project” on page 104
- “Adding a Pre-Build Event to Create a Federation” on page 106
- “Defining a Custom Build Rule to Process DDL Files” on page 107
- “Adding Generated Files” on page 109
- “Setting the Includes Search Path” on page 110
- “Setting Link Properties” on page 110
- “Building and Executing the Application” on page 111

Preparing to Create an Objectivity/C++ Project

Before you create your new project, perform the following steps:

1. List any existing DDL files that define the persistence-capable classes for your application. Similarly, make a preliminary list of planned DDL files that are not yet created. (You can create these files later with Objectivity/Assist or with a text editor.) For information about DDL files and the files that you will generate from them, see *Objectivity/C++ Data Definition Language*.
2. Decide how you want to identify the federated database in which your application will store its persistent objects. You should:
 - Choose the system name (*fdSysName*) to be used. The system name will be specified as the base filename for the federated database’s boot file and system-database file.
 - Choose an integer identifier (*fdId*) for the federated database.For complete information about federated databases, see Chapter 4, “Federated Database Tasks,” in *Objectivity/DB Administration*.
3. If you have not done so already, set up the Visual C++ IDE to work with Objectivity/C++; see “Setting Up the Visual C++ IDE” on page 31.

Creating a New Objectivity/C++ Project

The steps in this section organize your project files similar to the provided Objectivity/C++ sample applications (see page 117).

NOTE If you change this recommended organization, you may have to adjust some of the steps in “Adding a Pre-Build Event to Create a Federation” on page 106.

1. In the Visual C++ IDE, click **File > New > Project**.
2. In the New Project dialog:
 - a. Click **Visual C++** in the left pane.
 - b. In the center pane, click the desired project template—for example, Win32 Console Application.
 - c. Specify the name and location for the project. This step creates a storage folder with the project’s name in the specified location.
 - d. Uncheck **Create directory for solution**.
 - e. Follow any additional prompts, and click **OK**.
3. In the Solution Explorer, right-click your project and click **Add > New Filter** to add the following filter:

Persistent Classes	Project subfilter for files associated with your project’s persistence-capable classes.
--------------------	---

4. In the Solution Explorer, right-click your project’s Persistent Classes filter and click **Add > New Filter** to add each of the following subfilters:

DDL Files	Subfilter for DDL files (.ddl).
DDL Generated Files	Subfilter for files generated from DDL files (.h, _ref.h, and _ddl.cpp).
Source	Subfilter for any C++ source files (.cpp) implementing persistence-capable classes.

5. In Windows Explorer, create subfolders for organizing project files in storage:
 - a. Find your new project's storage folder.
 - b. Within the project's storage folder, create the storage folders listed below:

<code>data</code>	Storage folder for files associated with your project's federated database.
<code>ddlFiles</code>	Storage folder for your project's DDL files (.ddl) and all accompanying generated files (.h, _ref.h, and _ddl.cpp).
<code>src</code>	Storage folder for the C++ source files (.cpp) that implement your project's persistence-capable classes. Other source files (.h and .cpp) can optionally be stored here, as well.

- c. Move any existing DDL files into the `ddlFiles` storage folder.
6. Add each planned DDL file to your project:
 - a. In the Solution Explorer, right-click the `DDL Files` subfilter of the `Persistent Classes` filter and click **Add > New Item**.
 - b. In the Add New Item dialog, click **Browse** and select the `ddlFiles` storage folder.
 - c. In the left pane, click **Visual C++**. In the center pane, click the **C++ File (.cpp)** template.
 - d. In the Name box, enter the name of the planned DDL file, including the `.ddl` filename extension, and click **Add**.
7. Add any existing DDL files to your project:
 - a. In the Solution Explorer, right-click the `DDL Files` subfilter of the `Persistent Classes` filter and click **Add > Existing Item**.
 - b. In the Add Existing Item dialog, browse to the `ddlFiles` storage folder.
 - c. Set Files of type to display all files.
 - d. Select the DDL files in the `ddlFiles` storage folder and click **Add**.
 - e. If a dialog asks if you want to create a new rule file, click **No**.

Adding a Pre-Build Event to Create a Federation

The steps in this section set up your Visual C++ project to create a federated database when you build the project. These steps assume you have created a project and organized its storage folders as recommended in the preceding section.

NOTE If you changed the recommended organization, you may have to adjust some of the steps for adding a pre-build event.

1. In the Solutions Explorer, right-click the project, and click **Properties**.
2. In the left pane, click **Configuration Properties**, then in the Configuration box, click **All Configurations**.
3. If necessary, click to expand **Configuration Properties**; then, expand **Build Events** and select its **Pre-Build Event** property page.

4. Click the **Command Line** property and enter:

```
if exist "$(ProjectDir)data\fdSysName.boot"
    oocleanup -local "$(ProjectDir)data\fdSysName.boot"
if exist "$(ProjectDir)data\fdSysName.boot"
    oodeletefd -f "$(ProjectDir)data\fdSysName.boot"
del "$(ProjectDir)data\*.DB
oonewfd -fdfilepath "$(ProjectDir)data\fdSysName.fdb"
        -lockserverhost $(COMPUTERNAME)
        -fdnumber fdId "$(ProjectDir)data\fdSysName.boot"
```

Note: *fdSysName* is the chosen system name of the federated database, and *fdId* is a unique integer identifier for the federated database.

Note: Put each command and its options on a single line. Arguments to Objectivity/DB tools must be enclosed in double-quotation marks if they specify paths that contain spaces.

For more information about the `oocleanup`, `oodeletefd`, or `oonewfd` tools, see *Objectivity/DB Administration*.

5. Click the **Description** property and enter:
Creating Objectivity/DB Federated Database
6. In the Property Pages dialog, click **OK**.

Defining a Custom Build Rule to Process DDL Files

The steps in the following subsections set up your Visual C++ project to run the DDL processor as necessary when you build your project. These steps assume you have created a project and organized its filters as recommended in the preceding section.

NOTE If you changed the recommended organization, you may have to adjust some of the steps for defining custom build rules.

Visual C++ 8.0/9.0

1. In the Solutions Explorer, select all of the DDL files in the DDL Files subfilter of the project's Persistent Classes filter. (Hint: Hold the Control key while selecting each file.)
2. Right-click the selection and click **Properties**.
3. In the Configuration box of the Property Pages dialog, click **All Configurations**.
4. If necessary, click to expand **Configuration Properties**; then click **Custom Build Step** and select its **General** property page.
5. Click the **Command Line** property and enter:

```
cd $(InputDir)
oodd1x "$(InputName)".ddl ..\data\fdSysName.boot
cd $(ProjectDir)
```

Note: *fdSysName.boot* is the name of the federated-database boot file. For more information about the *oodd1x* command, see *Objectivity/C++ Data Definition Language*.

Note: Arguments to the DDL processor must be enclosed in double-quotation marks if they specify paths that contain spaces.

6. Click the **Description** property and enter:
Building DDL File \$(InputPath)
7. Click the **Outputs** property and enter:
\$(InputDir)\$(InputName)_ref.h
\$(InputDir)\$(InputName)_ddl.cpp
\$(InputDir)\$(InputName).h
8. Click **Additional Dependencies** and enter the boot-file name:
data\fdSysName.boot
9. In the Property Pages dialog, click **OK**.

Visual C++ 10.0

1. In the Solutions Explorer, select all of the DDL files in the `DDL Files` subfilter of the project's `Persistent Classes` filter. (Hint: Hold the Control key while selecting each file.)
2. Right-click the selection and click **Properties**.
3. Click **Item Type**, click **Custom Build Tool**, then click **Apply**.
4. In the Configuration box of the Property Pages dialog, click **All Configurations**.
5. If necessary, click to expand **Configuration Properties**; then click **Custom Build Tool** and select its **General** property page.
6. Click the **Command Line** property and enter:

```
cd $(ProjectDir)ddlFiles\  
oodd1x %(FullPath) ..\data\fdSysName.boot  
cd $(ProjectDir)
```

Note: `fdSysName.boot` is the name of the federated-database boot file. For more information about the `oodd1x` command, see *Objectivity/C++ Data Definition Language*.

Note: Arguments to the DDL processor must be enclosed in double-quotation marks if they specify paths that contain spaces.
7. Click the **Description** property and enter:

```
Building DDL File %(Filename)
```
8. Click the **Outputs** property and enter:

```
%(Filename)_ref.h  
%(Filename)_ddl.cpp  
%(Filename).h
```
9. Click **Additional Dependencies** and enter the boot-file name:

```
..\data\fdSysName.boot
```
10. In the Property Pages dialog, click **OK**.

Adding Generated Files

The steps in this section use your project to create a federated database and generate a set of C++ source files from your DDL files. You then add the generated files to your project.

NOTE Before performing the steps in this section, you must add persistence-capable class definitions to the DDL files referenced by your project; for details about defining persistence-capable classes, see *Objectivity/C++ Data Definition Language*.

1. In the Solution Explorer, right-click the project and click **Build**.
2. Add the generated header and implementation files to your project:
 - a. In the Solution Explorer, right-click the `DDL Generated Files` subfilter of the project's `Persistent Classes` filter and click **Add > Existing Item**.
 - b. In the Add Existing Item dialog, browse to the `ddlFiles` storage folder if necessary.
 - c. Select all of the `.h` and `.cpp` files in the `ddlFiles` storage folder and click **Add**.
3. Add any planned source (`.cpp`) files for implementing the member functions of your persistence-capable classes:
 - a. In the Solution Explorer, right-click the `Source` subfilter of the `Persistent Classes` filter and click **Add > New Item**.
 - b. In the Add New Item dialog, select the **C++ File (.cpp)** template in order to add the planned source (`.cpp`) files.
 - c. In the Add New Item dialog, click **Browse** and select the `src` storage folder.
 - d. In the Name box, enter the name of the planned implementation file, and click **Add**.
4. Add any existing source (`.cpp`) files for implementing the member functions of your persistence-capable classes:
 - a. In the Solution Explorer, right-click the `Source` subfilter of the `Persistent Classes` filter and click **Add > Existing Item**.
 - b. In the Add Existing Item dialog, browse to the `src` storage folder.
 - c. Select the implementation files in the `src` storage folder and click **Add**.
5. Add any remaining C++ source files (or references to them) to your project as you normally would.

Setting the Includes Search Path

The steps in this section ensure that generated header files can be found during compilation.

1. In the Solution Explorer, right-click your project and click **Properties**.
2. Click **Configuration Properties** in the left pane, then click **All Configurations** in the Configuration box.
3. If necessary, click to expand **Configuration Properties**, expand **C/C++**, and select its **Precompiled Headers** property page.
4. Specify that precompiled headers are not used:
 - *Visual C++ 8.0 and Visual C++ 9.0.* Select **Not Using Precompiled Headers** in Create/Use Precompiled Headers.
 - *Visual C++ 10.0.* Select **Not Using Precompiled Headers** in Precompiled Header.
5. Under **C/C++**, select the **General** property page.
6. In the Additional Include Directories text box, enter the pathname to your project's `ddlFiles` storage folder.
7. In the Property Pages dialog, click **OK**.

Setting Link Properties

You can set Visual C++ IDE properties as shown in the following steps to trigger automatic linking of Objectivity/DB libraries; see also "Automatic Linking of Objectivity/DB Libraries" on page 92.

- To link dynamic using the MFC:
 - a. In the Solution Explorer, right-click your project and click **Properties**.
 - b. In the Configuration box of the Property Pages dialog, click **All Configurations**.
 - c. If necessary, click to expand **Configuration Properties**; then, select the **General** property page.
 - d. In the Project Defaults section, click the **Use of MFC** property and choose the following value:
Use MFC in a Shared DLL
 - e. In the Property Pages dialog, click **OK**.

- To link dynamic without using the MFC:
 - a. In the Solution Explorer, right-click your project and click **Properties**.
 - b. In the Property Pages dialog, click **All Configurations** in the Configurations list.
 - c. If necessary, click to expand **Configuration Properties**, then expand **C/C++** and select the **Code Generation** property page.
 - d. Click the **Runtime Library** property and choose the following value:
Multi-threaded DLL (/MD)
 - e. In the Property Pages dialog, click **OK**.

Building and Executing the Application

The steps in this section automatically create a federated database, run the DDL processor, and build the application.

1. Right-click your project and click **Rebuild**.
2. Click **Debug > Start Without Debugging**.

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 *Objectivity/DB Administration*.
- Run your application in debug mode for data verification and event tracing; see *Objectivity/C++ Programmer’s Guide*.

In either case, you must first prepare your application for debugging, as described in the following subsection. The remaining subsections describe how to view persistent objects from the Visual C++ debugger.

Preparing to Debug an Application

1. In the Solution Explorer, right-click your project and click **Properties**.
2. If necessary, click **Configuration Properties** in the left pane.
3. Click **Debug** in the Configuration box and select the appropriate Windows Platform in the Platform box.
4. Expand **C/C++** in the left pane and click the **General** property page.
5. Click the **Debug Information Format** property and choose the following value: **Program Database (/ZI)**
6. Expand **Linker** in the left pane and click **Debugging**.
7. Click the **Generate Debug Info** property and choose the following value: **Yes (/DEBUG)**
8. In the Property Pages dialog, click **OK**.
9. Link your application to the debug-compatible Objectivity/DB import library; see “Automatic Linking of Objectivity/DB Libraries” on page 92.

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 *Objectivity/DB Administration*.

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.

B

Java Application Development

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

This appendix provides information about:

- Java command line options

Java Command Line Options

64-bit Architectures

To run an application on a Windows architecture that has the 64-bit addressing mode, you must specify the `-d64` option. This uses the 64-bit data model of the Java virtual machine instead of the default 32-bit data model.

C

Sample Applications

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

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

Table C-1: Location of Samples

Samples for This Programming Interface	Are in Subfolders of This Folder
Objectivity/C++	<i>installDir\samples\cxx</i>
Objectivity for Java	<i>installDir\samples\java</i>
Objectivity/.NET for C#	<i>installDir\samples\NET-csharp</i>

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

Overview of Sample Applications

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

Table C-2: Tasks Performed by Sample Applications

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

Sample Federated Databases

In some cases, the same federated database is accessed by corresponding applications for more than one programming-language interface. The files for such federated databases are in subfolders of *installDir\samples\data*.

NOTE The Objectivity/.NET for C# samples do not make use of the *installDir\samples\data* directory. Instead, each of these samples has its own local *data* directory.

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++, 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 the Control Panel, open **Add/Remove Programs**.
(*Windows 7*) In the Control Panel, click on **Programs and Features**.
2. Select the Objectivity product and version you want to uninstall.
3. Click **Remove** (or **Uninstall**).

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

Troubleshooting an Application

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

Federated Database Does Not Open

Solutions:

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

Lock Server Not Running

Solution:

- Run `oolockmon` to check whether a lock server is running. If necessary, run `oolockserver` to start 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 network timeout period by setting the `OO_RPC_TIMEOUT` environment variable to the desired number of seconds (greater than the default of 25 seconds).
- If you are accessing files on a UNIX host running NFS, consider decreasing the NFS data packet size by setting the `OO_NFS_MAX_DATA` environment variable to the desired number of bytes (less than the default of 8192 bytes).

Index

A

About This Release shortcut 18

Active Schema

for C++

installing 37

linking application 96

preparing to use 38

system requirements 37

for Java

installing 51

preparing to use 52

release files 52

system requirements 51

adjusting data packet size 27

Advanced Multithreaded Server (see AMS)

AMS

permissions 26

stopped 25

used with Objectivity/HA 79

verifying installation 24

application

developing C++ 91

linking 91

programming issues 102

Assist 22

configuring your own Eclipse for 23

shortcut 18

software requirements 14

starting 22

workspace directory 22

automatic linking 92

autonomous partitions 79

C

C++ application development 91

CLASSPATH environment variable 47

Customer Support 11

D

Data Definition Language (see DDL)

data packet size 27

data replication option (see Objectivity/HA)

data source

adding for federated database 73, 74

database images 79

DDL files 29

DDL processor 29

setting up for C++ development 107

debugging

viewing the value of an object 112

declspec(thread) extension 101

decorating names 102

demo applications

Interactive SQL++ 68

Objectivity for Java 48, 117

Objectivity/C++ 34, 117

Objectivity/.NET for C# 43

Objectivity/ODBC 76

Objectivity/Python 61

developing

C++ applications 91

Dharma/SQL 63

DLL

- exporting persistent data 100
- loading 101
- Objectivity/AS 96
- Objectivity/C++ persistent collections 96, 99
- Objectivity/DB 92
- Objectivity/IPLS 89, 98, 99
- Objectivity/PQE 86, 97, 98

download installer 15

DRO (see Objectivity/HA)

dynamic link import library (see import library)

dynamic link library (see DLL)

E**Eclipse Platform**

- configuring for Assist 23

Eclipse Rich Client Platform 22

environment variable settings 20

environment variables

- CLASSPATH 47
- include 20
- lib 20
- OO_FD_BOOT 123
- OO_NFS_MAX_DATA 27, 124
- OO_RPC_TIMEOUT 124
- OO_SQL_DIR 64
- path 20
- PYTHONPATH 60

ENVY/Developer

- requirements 53
- setting up 56

errors from running an application 123

exporting persistent data 100

F

fault tolerant option (see Objectivity/HA)

federated database 13

- adding data sources 73, 74
- license in 21

FTO (see Objectivity/HA)

H

HA abbreviation 10

high availability (see Objectivity/HA)

hostname for ODBC server 75

I

import library 99

- Objectivity/AS 96
- Objectivity/DB 92

include environment variable 20

installDir 19

installing

- Active Schema for C++ 37
- Active Schema for Java 51
- Objectivity for Java 45
- Objectivity/C++ 29
- Objectivity/DB 13
- Objectivity/DDL 29
- Objectivity/HA 79
- Objectivity/IPLS 87
- Objectivity/.NET for C# 41
- Objectivity/ODBC 71
- Objectivity/PQE 83
- Objectivity/Python 59
- Objectivity/Smalltalk for VisualWorks 53
- Objectivity/SQL++ 63

Interactive SQL++

- defined 63
- sample application 68
- testing 68

interoperating between releases 17

IPLS abbreviation 10

L

lib environment variable 20

library

- automatic linking 92
- compatibility 94
- dynamic link (see DLL)
- import (see import library)

license 21

linking

- automatic 92
- compatible libraries 94
- lock-server performance-monitoring program 99
- order 93
- user application 91
- user-created DLL 101
- Visual C++ IDE options 110
- Visual C++ runtime libraries 93

loading DLL 101**lock server**

- in-process 87
- performance-monitoring program, linking 99
- permissions 26
- stopped 25
- verifying installation 24

M**mangling names 102****memory-checking 113****monitoring lock-server performance 99****N****Network File System (NFS) 27**

- data packet size 27, 124

network timeout period, setting 124**O****Objectivity Books shortcut 18****Objectivity for Java**

- compiler requirements 45
- installing 45
- preparing to use 46
- release files 47
- sample applications 48, 117
- testing 48

Objectivity for Java Books shortcut 18, 47**Objectivity for Java with Java Connection Architecture (JCA) 45****Objectivity Network Services**

- shortcut 18
- tool 24, 66

Objectivity servers

- AMS 24
- installed as services 16, 24, 66
- lock server 24
- ODBC server 66
- query server 85
- verifying installation 24, 66, 85

Objectivity/.NET for C# shortcut 18**Objectivity/Assist (see Assist)****Objectivity/C++**

- compiler requirements 29
- debugging application 112
- installing 29
- linking application 91
- persistent collections libraries 95
- preparing to use 30
- programming issues 102
- release files 31
- sample applications 34, 117
- system requirements 29, 45
- testing 34

Objectivity/DB

- installing 13
- preparing to use 17
- release files 19
- shortcuts 18
- system requirements 14
- upgrading existing federated databases 17

Objectivity/DB Active Schema

(see Active Schema)

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

- installing 29
- preparing to use 30
- release files 31
- system requirements 29, 45
- testing 34

Objectivity/DRO (see Objectivity/HA)

Objectivity/FTO (see Objectivity/HA)

Objectivity/HA 79

- installing 79
- preparing to use 80
- release files 81
- system requirements 79

Objectivity/IPLS

- installing 87
- loading DLL 89, 98
- preparing to use 88
- release files 89
- system requirements 87

Objectivity/.NET for C#

- compiler requirements 41
- installing 41
- preparing to use 42
- release files 43
- sample applications 43
- system requirements 41
- testing 43

Objectivity/ODBC

- adding data sources 73, 74
- configuring TCP/IP 75
- installing 71
- preparing to use 73
- release files 73
- sample client application 76
- software requirements 72
- system requirements 71
- testing 76

Objectivity/Parallel Query Engine (see Objectivity/PQE)

Objectivity/PQE 83

- installing 83
- loading DLL 86, 97
- preparing to use 84
- query server 85
 - checking and setting up 85
- release files 84
- system requirements 83

Objectivity/Python 59

- installing 59
- preparing to use 60
- release files 61

- sample applications 61
- system requirements 59
- testing 61

Objectivity/Smalltalk for VisualWorks

- installing 53
- preparing to use 54
- release files 55
- system requirements 53
- testing 57

Objectivity/SQL++

- Dharma/SQL version embedded 63
- installing 63
- Interactive SQL++
 - defined 63
 - testing 68
- log directory 67
- ODBC server
 - defined 63
 - permissions 67
 - registering port number 75
 - setting up 66
 - specifying hostname 75
 - stopped 66
 - testing 69
 - verifying installation 66
- preparing to use 64
- release files 65
- supported ODBC standard 63
- system requirements 63
- testing
 - Interactive SQL++ 68
 - ODBC server 69

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

ODBC

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

ODMG abbreviation 10

online books

- location 19

oo.h include file 92

OO_FD_BOOT environment variable 123

OO_NFS_MAX_DATA environment variable 27, 124

OO_RPC_TIMEOUT environment variable
124

OO_SQL_DIR environment variable 64

ooas.lib 96

ooasd.lib 96

oocleanup.exe 102

ooco.lib 95

oocod.lib 95

oodbi.lib 92

oodbid.lib 92

oodbxx.dll 92

oodbxxd.dll 92

oolspmi.lib 99

oolspmid.lib 99

oosqlnw service 75

P

packet size 27

path environment variable 20

persistent collections
libraries for 95

port number
conflict 25, 67
registering, for ODBC server 75

PQE (Objectivity/PQE)

PQE abbreviation 10

predefined signal handler 102

preparing to use
Active Schema for C++ 38
Active Schema for Java 52
Objectivity for Java 46
Objectivity/C++ 30
Objectivity/DB 17
Objectivity/DDL 30
Objectivity/HA 80
Objectivity/IPLS 88
Objectivity/.NET for C# 42
Objectivity/ODBC 73
Objectivity/PQE 84
Objectivity/Python 60
Objectivity/Smalltalk for VisualWorks 54
Objectivity/SQL++ 64

project file
C++ sample application 34

Python Books shortcut 18

PYTHONPATH environment variable 60

Q

query server
checking 85

R

releases, interoperating between 17

RPC timeout error 27
(see also network timeout period)

S

sample applications
Interactive SQL++ 68
Objectivity for Java 48, 117
Objectivity/C++ 34, 117
Objectivity/.NET for C# 43
Objectivity/ODBC 76
Objectivity/Python 61

search path for Objectivity/DB tools 20

setting up
Eclipse Platform for Assist 23
Visual C++ IDE 31
VisualWorks 56
VisualWorks with ENVY/Developer 56

setup.exe 16

shortcuts
About This Release 18
Assist 18
Objectivity Books 18
Objectivity for Java Books 18, 47
Objectivity Network Services 18
Objectivity/.NET for C# 18
Python Books 18

solution file
for sample C# application 43

status 85

stopped Objectivity server 25, 66
system 64

T

TCP/IP

- configuring for Objectivity/ODBC 75
- installing 26
- services file 76

troubleshooting applications 123

typographical conventions 10

U

uninstalling Objectivity products 121

upgrading federated databases
for Objectivity/DB 17

user-created DLL, linking 101

V

Visual C++ IDE

- defining custom build rules 107
- developing Objectivity/DB applications
103
- setting link options 110
- setting up 31

Visual C++ name decoration 102

Visual Studio 41

VisualWorks

- requirements 53
- setting up 56

W

WebLogic 45

WebSphere 45

Windows 2003, abbreviation 14

Windows 2008, abbreviation 14

Windows XP x64, abbreviation 14

Windows XP, abbreviation 14

Windows-specific development issues
Objectivity for Java 115