Pro*C/C++ Precompiler

Getting Started

Release 9.0.1 for Windows

June 2001

Part No. A90167-01

ORACLE

Pro*C/C++ Precompiler Getting Started, Release 9.0.1 for Windows

Part No. A90167-01

Copyright © 1994, 2001, Oracle Corporation. All rights reserved.

Contributors: Riaz Ahmed, Eric Belden, Janis Greenberg, Sharon Castledine, Joseph Garcia, Lisa Giambruno, Neeraj Gupta, Bernie Harris, Ana Hernandez, Mark Kennedy, Robert Knecht, Shiva Prasad, Ali Shehade, Helen Slattery, Jeff Stein, Nicole Sullivan, Janice Wong, Martha Woo, Ravi Gooty

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark, and Oracle9*i*, PL/SQL, and Pro*C/C++ are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Send Us Your Comments

Pro*C/C++ Precompiler Getting Started, Release 9.0.1 for Windows

Part No. A90167-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- E-mail: ntdoc_us@oracle.com
- FAX - (650) 506-7365 Attn: Oracle Database for Windows Documentation
- Postal service:
  Oracle Corporation
  Oracle Database for Windows Documentation Manager
  500 Oracle Parkway, Mailstop 1op6
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services. Contact information for Oracle Support Services is available at this Web site:

`http://www.oracle.com/support/`

Preface

This guide provides introductory information for the Pro*C/C++ precompiler running on Microsoft Windows operating systems.

This preface contains these topics:

Audience

*Pro*C/C++ Precompiler Programmer's Guide* is intended for anyone who wants to use Pro*C/C++ to perform the following tasks:

- Embed SQL statements in a C or C++ program.
- Build Oracle database applications with Pro*C/C++.

To use this document, you need to know:

- Commands for deleting and copying files and the concepts of the search path, subdirectories, and path names.
- Microsoft Windows NT, Windows 95/98, or Windows 2000 operating system.
- Microsoft Visual C++ version 5.0 or higher.

Organization

This document contains:

Chapter 1, "Introducing Pro*C/C++"

Describes Pro*C/C++, the Oracle programmatic interface for the C and C++ languages running on Microsoft Windows NT and Windows 95/98 operating systems.

Chapter 2, "Using Pro*C/C++"

Explains how to create and precompile a project. Also describes the Pro*C/C++ graphical user interface, from which you execute commands with Windows menus and icons or with keyboard equivalents, and using Pro*C/C++ at the command line.

Chapter 3, "Sample Programs"

Describes how to build Oracle database applications with Pro*C/C++ using the sample programs that are included with this release, and provides an overview of how to build multi-threaded applications.

Appendix A, "Integrating Pro*C/C++ into Microsoft Visual C++"

Describes how to integrate Pro*C/C++ into the Microsoft Visual C++ integrated development environment.

Related Documentation

For more information, see these Oracle resources:

- Oracle9i Database installation guide for Windows
- Oracle9i Database release notes for Windows
- *Oracle9i Database Administrator's Guide for Windows*
- *Oracle Enterprise Manager Administrator's Guide*
- *Oracle9i Net Services Administrator's Guide*
- *Oracle9i Real Application Clusters Concepts*
- *Oracle9i Database New Features*
- *Oracle9i Database Concepts*
- *Oracle9i Database Reference*
- *Oracle9i Database Error Messages*
- *Pro*C/C++ Precompiler Programmer's Guide*

In North America, printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com/`

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

`http://www.oraclebookshop.com/`

Other customers can contact their Oracle representative to purchase printed documentation.

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

`http://technet.oracle.com/membership/index.htm`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://technet.oracle.com/docs/index.htm`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples
- Conventions for Windows Operating Systems

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width font)`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.

Convention	Meaning	Example
`lowercase monospace (fixed-width font)`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase monospace (fixed-width font) italic`	Lowercase monospace italic font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run `U`*`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`

Convention	Meaning	Example
. . .	Horizontal ellipsis points indicate either: ■ That we have omitted parts of the code that are not directly related to the example ■ That you can repeat a portion of the code	`CREATE TABLE ... AS subquery;` `SELECT col1, col2, ... , coln FROM employees;`
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	`acctbal NUMBER(11,2);` `acct CONSTANT NUMBER(4) := 3;`
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/system_password` `DB_NAME = database_name`

UPPERCASE	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Conventions for Windows Operating Systems

The following table describes conventions for Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Choose Start >	How to start a program. For example, to start Oracle Database Configuration Assistant, you must click the Start button on the taskbar and then choose Programs > Oracle - *HOME_NAME* > Database Administration > Database Configuration Assistant.	Choose Start > Programs > Oracle - *HOME_NAME* > Database Administration > Database Configuration Assistant
`C:\>`	Represents the Windows command prompt of the current hard disk drive. Your prompt reflects the subdirectory in which you are working. Referred to as the command prompt in this guide.	`C:\oracle\oradata>`
HOME_NAME	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*`HOME_NAME`*`TNSListener`

Convention	Meaning	Example
ORACLE_HOME and *ORACLE_BASE*	In releases prior to 8.1, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory that by default was: ■ `C:\orant` for Windows NT ■ `C:\orawin95` for Windows 95 ■ `C:\orawin98` for Windows 98 or whatever you called your Oracle home. In this Optimal Flexible Architecture (OFA)-compliant release, all subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is `C:\oracle`. If you install release 9.0 on a computer with no other Oracle software installed, the default setting for the first Oracle home directory is `C:\oracle\ora90`. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. See *Oracle9i Database Getting Started for Windows* for additional information on OFA compliances and for information on installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\admin` directory.

Documentation Accessibility

Oracle's goal is to make our products, services, and supporting documentation accessible to the disabled community with good usability. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

What's New in Pro*C/C++?

The following sections describe the new features in Oracle Pro*C/C++:

- Oracle9i Release 1 (9.0.1) New Features in Pro*C/C++

Oracle9*i* Release 1 (9.0.1) New Features in Pro*C/C++

This section contains these topics:

- **Using Oracle9*i* on Windows 2000**

  There are some differences between using Oracle9*i* on Windows 2000 and Windows NT 4.0.

  See Also: *Oracle9i Database Getting Started for Windows*

1

Introducing Pro*C/C++

This chapter describes Pro*C/C++, the Oracle programmatic interface for the C and C++ languages running on Window operating systems. Pro*C/C++ enables you to build Oracle database applications in a Win32 environment.

This chapter contains these topics:

- What is Pro*C/C++?
- Features
- Restrictions
- Directory Structure

> **Note:** See the *Pro*C/C++ Precompiler Programmer's Guide* for additional information.

What is Pro*C/C++?

Pro*C/C++ precompiler enables you to create applications that access your Oracle database whenever rapid development and compatibility with other systems are your priorities.

The Pro*C/C++ programming tool enables you to embed Structured Query Language (SQL) statements in a C or C++ program. The Pro*C/C++ precompiler translates these statements into standard Oracle runtime library calls, then generates a modified source program that you can compile, link, and run in the usual way.

Features

Pro*C/C++ supports the following features:

- Remote access with Oracle Net or local access to Oracle databases
- Embedded PL/SQL blocks
- Bundled database calls, which can provide better performance in client/server environments
- Full ANSI compliance for embedded SQL programming
- PL/SQL version 9.0 and host language arrays in PL/SQL procedures
- Multi-threaded applications
- Full ANSI C compliance
- Microsoft Visual C++ support, version 6.0 for 32-bit applications

Note: Borland C++ is no longer supported.

Restrictions

Pro*C/C++ does not support 16-bit code generation.

Directory Structure

Installing Oracle software creates a directory structure on your hard drive for the Oracle products. A main Oracle directory contains the Oracle subdirectories and files that are necessary to run Pro*C/C++.

When you install Pro*C/C++, Oracle Universal Installer creates a directory called `\precomp` in the *`ORACLE_BASE`*`\`*`ORACLE_HOME`* directory. This subdirectory contains the Pro*C/C++ executable files, library files, and sample programs listed in Table 1–1, "precomp Directory Structure".

Table 1–1 precomp Directory Structure

Directory Name	Contents
`\admin`	Configuration files
`\demo\proc`	Sample programs for Pro*C/C++
`\demo\sql`	SQL scripts for sample programs
`\doc\proc`	Readme files for Pro*C/C++
`\help\proc`	Help files for Pro*C/C++
`\lib\msvc`	Library files for Pro*C/C++
`\mesg`	Message files
`\misc\proc`	Miscellaneous files for Pro*C/C++
`\public`	Header files

Note: The `\precomp` directory can contain files for other products, such as Pro*COBOL.

Known Problems, Restrictions, and Workarounds

Although all Windows operating systems allow spaces in filenames and directory names, the Oracle Pro*C/C++ and Oracle Pro*COBOL precompilers will not precompile files that include spaces in the filename or directory name. For example, do not use the following formats:

- `proc iname=test one.pc`
- `proc iname=d:\dir1\second dir\sample1.pc`

2

Using Pro*C/C++

This chapter explains how to create and precompile a project. It also describes the Pro*C/C++ graphical user interface, from which you execute commands with Windows menus and icons or with keyboard equivalents, and using Pro*C/C++ at the command line.

This chapter contains these topics:

> **Note:** See the *Pro*C/C++ Precompiler Programmer's Guide* for additional information about Pro*C/C++.

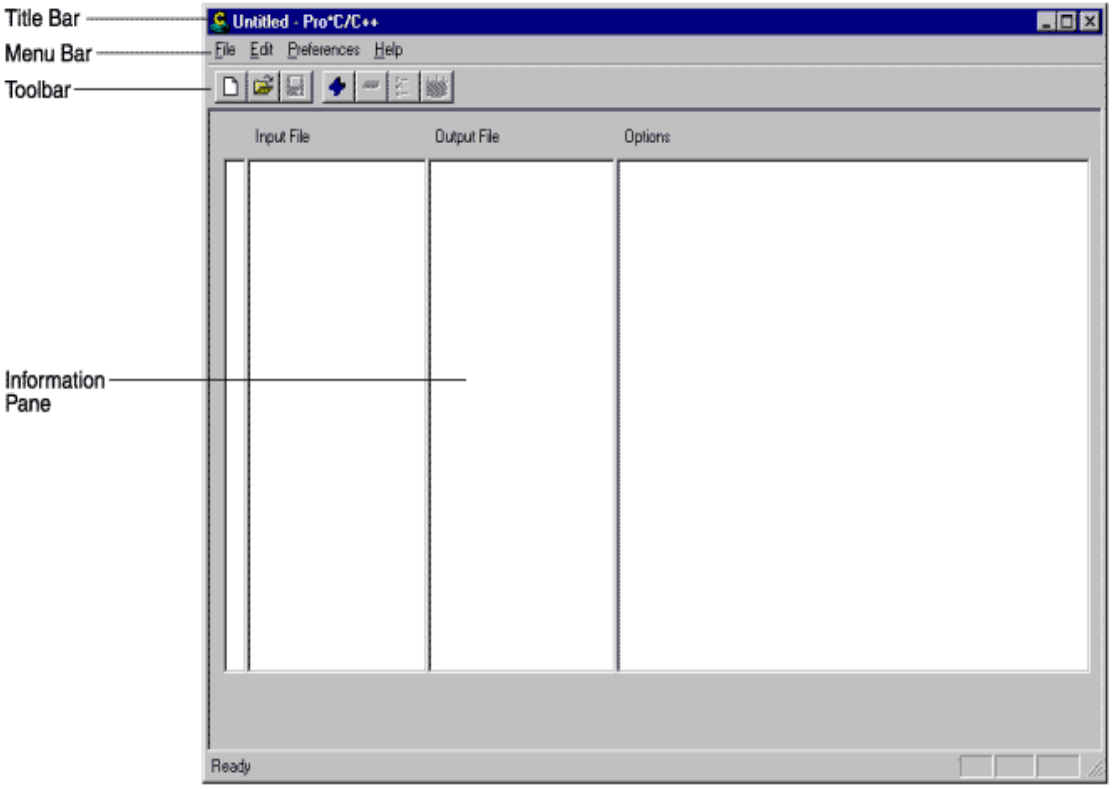
Using the Graphical User Interface

Before you follow the instructions for creating and precompiling a Pro*C/C++ project, you should familiarize yourself with the basic commands, dialog boxes, menus, and buttons of the Pro*C/C++ graphical user interface.

Starting Pro*C/C++ Graphical Interface

To start the graphical user interface, choose Start > Programs > Oracle - *HOME_NAME* > Application Development > Pro C_C++.

The Pro*C/C++ precompile environment contains five elements noted in the following illustration:



Title Bar

The title bar displays the name of the Pro*C/C++ project. If you have not assigned a name to the current project, the word “Untitled” appears instead.

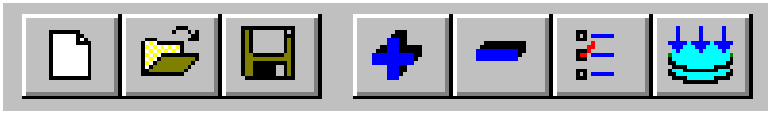
Menu Bar

The menu bar contains the following menus:

Menu	Description
File	Contains commands to create a new Pro*C/C++ project, open an existing Pro*C/C++ project, save the active Pro*C/C++ project under the same name or under a different name, specify a connect string to an Oracle database, precompile a Pro*C/C++ project, and exit the application.
Edit	Contains commands to add files to a Pro*C/C++ project, delete files from a Pro*C/C++ project, and display or change precompiler options.
Preferences	Contains commands to set the default file extension of output files.
Help	Contains the About Pro*C/C++ command, which displays the version number of the application and copyright information.

Toolbar

The toolbar enables you to execute commands by choosing a button:



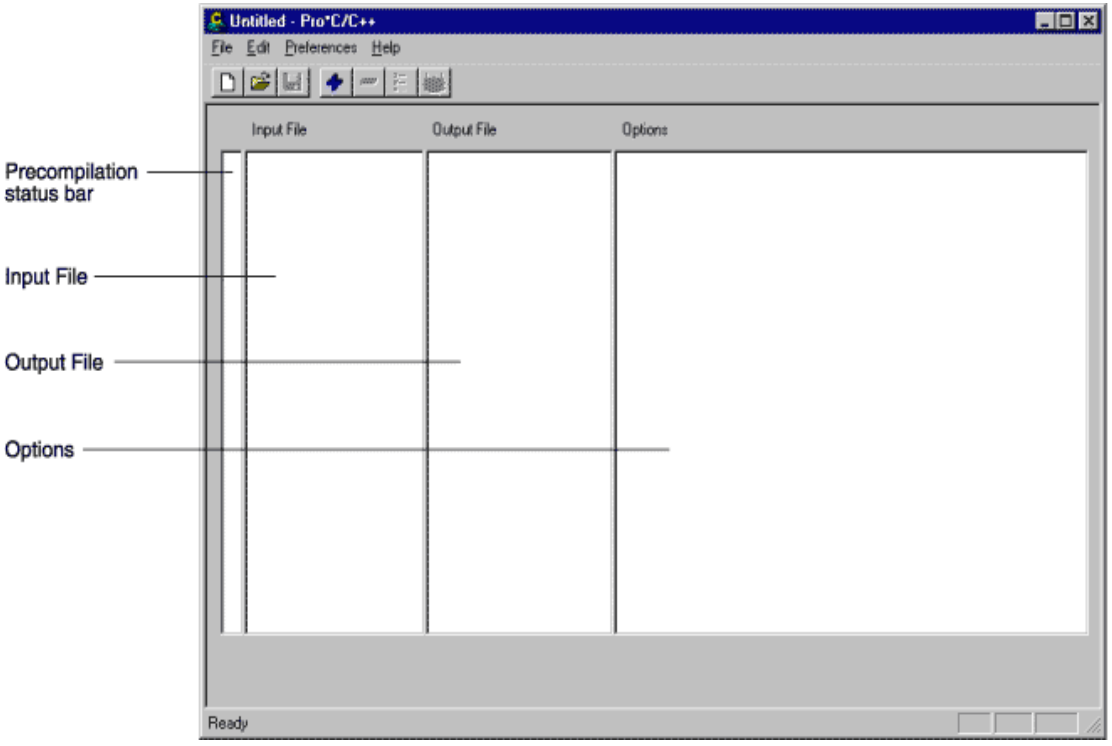
The following table describes the buttons in order, from left to right.

Button	Description
New	Create a new Pro*C/C++ project
Open	Open an existing Pro*C/C++ project
Save	Save the active Pro*C/C++ project under the same name
Add	Add files to a Pro*C/C++ project
Delete	Delete files from a Pro*C/C++ project

Button	Description
Options	Display or change precompiler options
Precompile	Precompile a Pro*C/C++ project

Information Pane

The information pane consists of four elements noted in the following illustration:



Element	Description
Precompilation Status Bar	Indicates whether the precompilation for a file was successful or unsuccessful.
Input File	Shows the files of a Pro*C/C++ project to be precompiled.
Output File	Shows the output files of a Pro*C/C++ project after precompilation.

Element	Description
Options	Displays precompile options that are different from the default options.

Look for one of the three status icons in the precompilation status bar after the precompile process is complete.

- A green check indicates that the file precompiled successfully.
- A yellow check indicates that the file precompiled successfully, but there are one or more warnings.
- A red X indicates that the file did not precompile successfully.

Double clicking a status icon opens the Precompilation Status dialog box. This dialog box provides detailed information on the reason for a warning or failure.

Status Bar

The status bar at the bottom of the window displays information about the progress of a precompilation. The status bar also identifies the purpose of a toolbar button or menu command when you place the mouse pointer over the toolbar button or menu command.

Creating and Precompiling a Pro*C/C++ Project

This section describes the steps involved in creating and precompiling a Pro*C/C++ project. After starting the Pro*C/C++ application, perform the following steps:

- Opening a Project
- Setting the Default Extension of Output Files
- Changing the Name of an Existing Input or Output File
- Adding Files to the Project
- Deleting Files from the Project
- Setting the Precompiler Options
- Specifying Database Connection Information
- Precompiling a Pro*C/C++ Project
- Checking the Results

- Fixing Errors
- Exiting Pro*C/C++

Opening a Project

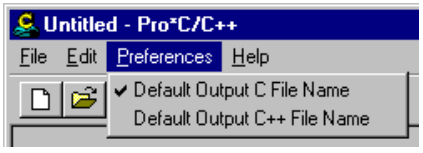
Pro*C/C++ opens only one project at a time. A project consists of one or more precompilable files. Project files have an extension of `.pre`.

- To create a new project, choose File > New Project.
- To open an existing project, choose File > Open Project.

> **Note:** A project created by a prior release cannot be opened by Oracle9*i*. It results in an Unexpected File Format error. You must recreate the project.

Setting the Default Extension of Output Files

Use the Preferences menu to determine the default extension of the output files.



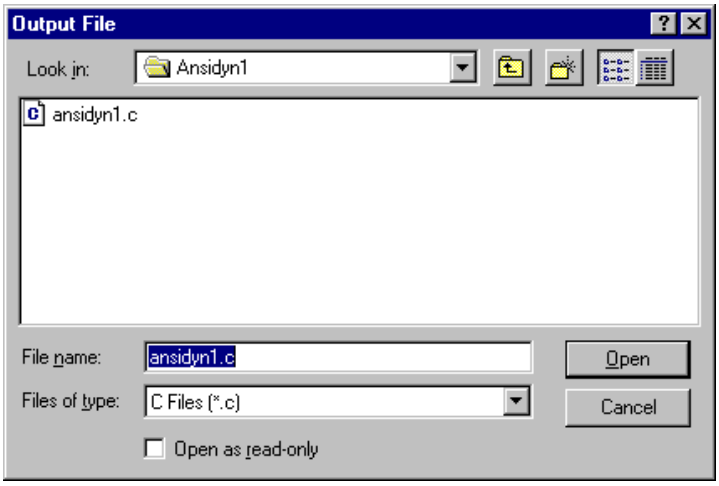
This setting only affects input files that you add later. An existing output filename will not change. However, you can change an existing output filename by double-clicking the output file and entering a new name.

- If you select Default Output C File Name, the default extension of the output files is `.c`.
- If you select Default Output C++ File Name, the default extension of the output files is `.cpp`.
- If you deselect both Default Output C File Name and Default Output C++ File Name, the Output File dialog box appears when you add an output file.
- Enter an output filename for the file selected. After you select or enter a filename, it appears in the Output File area of the information pane.

Changing the Name of an Existing Input or Output File

To change the name of an existing input or output file:

1. Double-click the filename in the Input File or Output File area of the information pane. The Input File or Output File dialog box appears.

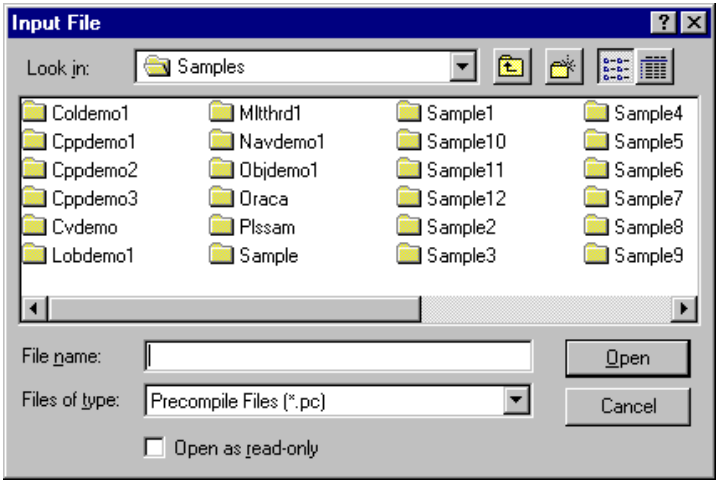


2. Replace the old filename with the new filename.
3. Choose Open.

Adding Files to the Project

To add files to the project:

1. Choose Edit > Add. The Input File dialog box appears.



2. Select one or more `.pc` files. Use the Ctrl key and the mouse to select files that are not adjacent.
3. Choose Open. The selected files appear in the information pane.

Deleting Files from the Project

If you need to, you can easily delete one or more files from the project.

To delete files from the project:

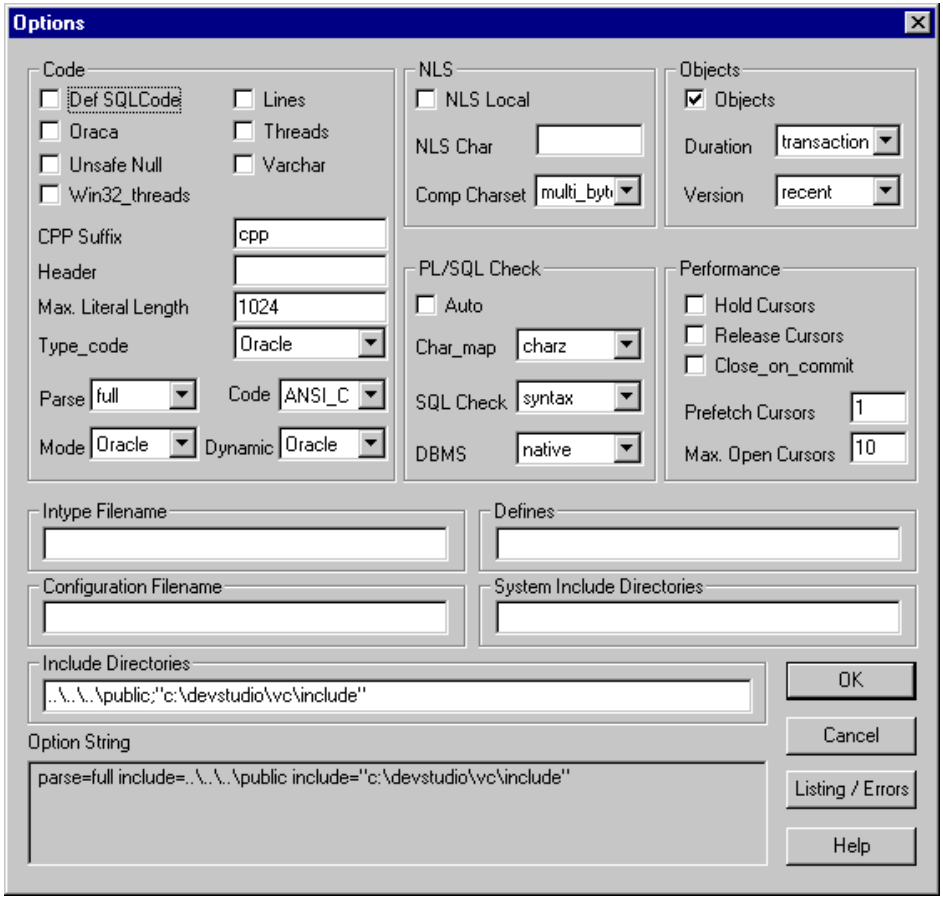
1. Highlight the file(s) in the information pane.
2. Choose Edit > Delete.
3. Choose Yes.

Setting the Precompiler Options

The Precompiler options enable you to control how resources are used, how errors are reported, how input and output are formatted, and how cursors are managed.

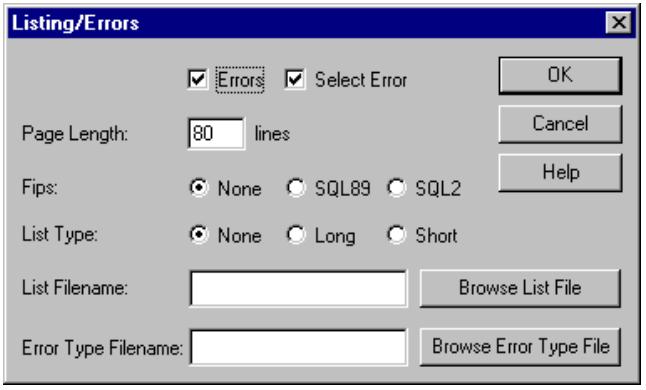
To set the precompile options:

1. Select one or more files in the Input File list.
2. Choose Edit > Options. The Options dialog box appears.



Default options are in effect for all newly added files. When you change an option's default setting, a description of the change appears in the Option String edit field at the bottom of the Options dialog box and in the Options area of the information pane. For additional information on options, see "Precompiler Options" on page 2-16.

3. To change the format of the output list file that the precompiler writes to disk, choose the Listing / Errors button. The Listing/Errors dialog box appears.



   The settings include the type of error information generated and the name of the list file.

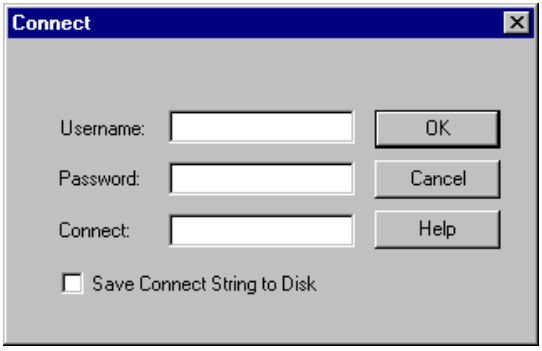
4. After you set the options in the Options dialog box, choose OK.

Specifying Database Connection Information

If you selected semantics or full for the SQL Check option in the Options dialog box, you may need to specify database connection information to the Oracle database. You do not need to connect to the Oracle database if every table referenced in a data manipulation statement or PL/SQL block is defined in a DECLARE TABLE statement.

To specify database connection information:

1. Choose File > Connect. The Connect dialog box appears.



2. Use this dialog box to specify database connection information prior to precompiling. No database connection is performed at this time. Only one set of database connection information can be specified for all files requiring semantic or full checking with SQLCHECK.
3. The Connect dialog box appears automatically at precompile time if you have not previously responded. Enter the username, the password, and the network service name (database alias). The network service name is not required for a local database.
4. If you want to save the connection information between Pro*C/C++ sessions, select the Save Connect String to Disk check box. If you do not select the check box, you must enter this information each time you precompile.
5. Choose OK.

Precompiling a Pro*C/C++ Project

You can precompile any number of files in the Input File list.

To precompile:

1. Select one or more files in the Input File list. You can use the Control key to highlight files that are not adjacent to each other (for example, the first and third files in a list).
2. Choose File > Precompile.

When precompiling is completed, the message in the dialog box indicates "`Precompiling Finished!`", and the Cancel button changes to OK.

3. Choose OK.

Note: Although Cancel does not interrupt the precompile for a file already in process, it does halt the precompile chain for remaining files.

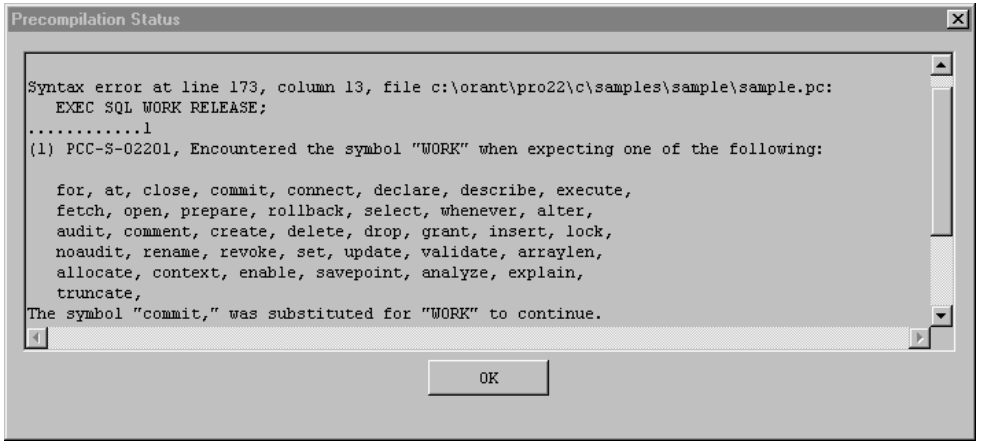
Checking the Results

Precompiling can result in success, success with warning(s), or failure. When precompiling is finished, check the precompilation status bar.

- A green check indicates that the file compiled successfully.
- A yellow check indicates that the file compiled successfully, but there are one or more warnings.
- A red X indicates that the file did not compile successfully.

Fixing Errors

If you see a yellow check or a red X, double-click the icon in status bar. The Precompilation Status dialog box appears. This dialog box lists warning messages or reasons why the precompilation failed. For example:



Switch to your development environment to fix the problem(s). After you correct the errors, precompile again.

Note: If you receive a PCC-S-02014 error (syntax error at line *num*, column *colnam*, file *name*), do the following:

- Copy the batch files `mod_incl.bat` and `add_newl.bat` from the *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\precomp\misc\proc` directory to the directory that contains the problematic INCLUDE file.
- Run `mod_incl.bat`.

Exiting Pro*C/C++

To exit Pro*C/C++, choose File > Exit. If your project changed in any way, you are prompted to save it.

Suggestion: If you want to keep an original file, as well as a version of the file with your changes, choose the Save As command. The Save command overwrites the previous version.

Using Pro*C/C++ at the Command Line

To precompile a file at the command line, enter the following command:

```
C:\> proc iname=filename.pc
```

where `filename.pc` is the name of the file. If the file is not in your current working directory, include the file's full path after the INAME argument.

Pro*C/C++ generates `filename.c`, which can be compiled by your C compiler.

Header Files

The *ORACLE_BASE**ORACLE_HOME*\precomp\public directory contains the Pro*C/C++ header files.

> **See Also:** See the *Pro*C/C++ Precompiler Programmer's Guide* for more information about oraca.h, sqlca.h, and sqlda.h.

Header File	Description
oraca.h	Contains the Oracle Communications Area (ORACA), which helps you to diagnose runtime errors and to monitor your program's use of various Oracle9*i* resources.
sql2oci.h	Contains SQLLIB functions that enable the Oracle Call Interface (OCI) environment handle and OCI service context to be obtained in a Pro*C/C++ application.
sqlapr.h	Contains ANSI prototypes for externalized functions that can be used in conjunction with OCI.
sqlca.h	Contains the SQL Communications Area (SQLCA), which helps you to diagnose runtime errors. The SQLCA is updated after every executable SQL statement.
sqlcpr.h	Contains platform-specific ANSI prototypes for SQLLIB functions that are generated by Pro*C/C++. By default, Pro*C/C++ does not support full-function prototyping of SQL programming calls. If you need this feature, include sqlcpr.h before any EXEC SQL statements in your application source file.
sqlda.h	Contains the SQL Descriptor Area (SQLDA), which is a data structure required for programs that use dynamic SQL Method 4.
sqlkpr.h	Contains K&R prototypes for externalized functions that can be used in conjunction with OCI.
sqlproto.h	The sqlproto.h header file was obsoleted in Pro*C/C++ release 8.0.3. Use sqlcpr.h instead of sqlproto.h. However, applications that were built using sqlproto.h can be created without modification: a dummy sqlproto.h file that includes sqlcpr.h has been provided in the *ORACLE_BASE**ORACLE_HOME*\precomp\public directory.

Library Files

The *ORACLE_BASE**ORACLE_HOME*\precomp\lib\msvc directory contains the library file that you use when linking Pro*C/C++ applications. The library file is called orasql9.lib.

Pro*C/C++ application program interface (API) calls are implemented in DLL files provided with your Pro*C/C++ software. To use the DLLs, you must link your application with the import libraries (.lib files) that correspond to the Pro*C/C++ DLLs. Also, you must ensure that the DLL files are installed on the computer that is running your Pro*C/C++ application.

Microsoft provides you with three libraries: libc.lib, libcmt.lib, and msvcrt.lib. The Oracle DLLs use the msvcrt.lib runtime library. You must link with msvcrt.lib rather than the other two Microsoft libraries.

Multithreaded Applications

Build multithreaded applications if you are planning to perform concurrent database operations.

Windows NT, Windows 2000, and Windows 95/98 schedule and allocate threads belonging to processes. A thread is a path of a program's execution. It consists of a kernel stack, the state of the CPU registers, a thread environment block, and a users stack. Each thread shares the resources of a process. Multithreaded applications use the resources of a process to coordinate the activities of individual threads.

When building a multithreaded application, make sure that your C/C++ code is reentrant. This means that access to static or global data must be restricted to one thread at a time. If you mix multithreaded and non-reentrant functions, one thread can modify information that is required by another thread.

The Pro*C/C++ precompiler automatically creates variables on the local stack of the thread. This ensures that each thread using the Pro*C/C++ function has access to a unique set of variables and is reentrant.

> **See Also:** See the *Pro*C/C++ Precompiler Programmer's Guide* for additional information on how to write multithreaded applications with Pro*C/C++.

Precompiler Options

This section highlights issues related to Pro*C/C++ for Windows platforms.

> **See Also:** See the "Precompiler Options" chapter of the *Pro*C/C++ Precompiler Programmer's Guide* for more information on the precompiler options.

Configuration File

A configuration file is a text file that contains precompiler options.

For this release, the system configuration file is called `pcscfg.cfg`. This file is located in the *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\precomp\admin` directory.

CODE

The `CODE` option has a default setting of `ANSI_C`. Pro*C/C++ for other operating systems may have a default setting of `KR_C`.

DBMS

`DBMS=V6_CHAR` is not supported when using `CHAR_MAP=VARCHAR2`. Instead, use `DBMS=V7`.

INCLUDE

For the Pro*C/C++ graphical user interface, use the Include Directories field of the Options dialog box to enter `INCLUDE` path directories. If you want to enter more than one path, separate each path with a semicolon, but do not insert a space after the semicolon. This causes a separate "`INCLUDE=`" string to appear in front of each directory.

For sample programs that precompile with `PARSE=PARTIAL` or `PARSE=FULL`, an include path of `c:\program files\devstudio\vc\include` has been added. If Microsoft Visual C++ has been installed in a different location, modify the Include Directories field accordingly for the sample programs to precompile correctly.

PARSE

The `PARSE` option has a default setting of `NONE`. Pro*C/C++ for other operating systems may have a default setting of `FULL`.

Using Pro*C/C++ with the Oracle XA Library

The XA Application Program Interface (API) is typically used to enable an Oracle database to interact with a transaction processing (TP) monitor, such as:

- BEA Tuxedo
- IBM Transarc Encina
- IBM CICS

You can also use TP monitor statements in your client programs. The use of the XA API is also supported from both Pro*C/C++ and OCI.

The Oracle XA Library is automatically installed as part of Oracle9*i* Enterprise Edition. The following components are created in your Oracle home directory:

Component	Location
`oraxa9.lib`	*ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\xa`
`xa.h`	*ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\demo`

Compiling and Linking a Pro*C/C++ Program with XA

To compile and link a Pro*C/C++ program with XA:

1. Precompile *`filename`*`.pc` using Pro*C/C++ to generate `filename.c`.
2. Compile *`filename`*`.c`, making sure to include *ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\xa` in your path.
3. Link *`filename`*`.obj` with the following libraries:

Library	Location
`oraxa9.lib`	*ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\xa`
`oci.lib`	*ORACLE_BASE*`\`*ORACLE_HOME*`\oci\lib\msvc`
`orasql9.lib`	*ORACLE_BASE*`\`*ORACLE_HOME*`\precomp\lib\msvc`

4. Run *`filename`*`.exe`.

XA Dynamic Registration

Oracle supports the use of XA dynamic registration. XA dynamic registration improves the performance of applications that interface with XA-compliant TP monitors.

For TP monitors to use XA dynamic registration with an Oracle database on Windows NT, you must add either an environmental variable or a registry variable to the Windows NT computer on which your TP monitor is running. See either of the following sections for instructions:

- Adding an Environmental Variable for the Current Session
- Adding a Registry Variable for All Sessions

Adding an Environmental Variable for the Current Session

Adding an environmental variable at the command prompt affects only the current session.

To add an environmental variable for the current session:

1. Go to the computer where your TP monitor is installed.
2. Enter the following at the command prompt:

   ```
   C:\> set ORA_XA_REG_DLL = vendor.dll
   ```

   where *vendor*`.dll` is the TP monitor DLL provided by your vendor.

Adding a Registry Variable for All Sessions

Adding a registry variable affects all sessions on your Windows NT computer. This is useful for computers where only one TP monitor is running.

To add a registry variable for all sessions:

1. Go to the computer where your TP monitor is installed.
2. Enter the following at the command prompt:

   ```
   C:\> regedt32
   ```

   The Registry Editor window appears.
3. Go to `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\HOME`*ID*.
4. Choose the Add Value option in the Edit menu.

   The Add Value dialog box appears.

5. Enter `ORA_XA_REG_DLL` in the Value Name field.
6. Select `REG_EXPAND_SZ` from the Data Type drop-down list box.
7. Choose OK.

   The String Editor dialog box appears.
8. Enter *`vendor`*`.dll` in the String field, where *`vendor`*`.dll` is the TP monitor DLL provided by your vendor.
9. Choose OK.

   The Registry Editor adds the parameter.
10. Choose Exit from the Registry menu.

    The registry exits.

XA and TP Monitor Information

Refer to the following for more information about XA and TP monitors:

- *Distributed TP: The XA Specification* (C193) published by the Open Group. See the Web site at:

  `http://www.opengroup.org/publications/catalog/tp.htm`

- The Open Group., 1010 El Camino Real, Suite 380, Menlo Park, CA 94025, U.S.A.
- Your specific TP monitor documentation

See Also: For more information about the Oracle XA Library and using XA dynamic registration, see *Oracle9i Application Developer's Guide - Fundamentals*.

3

Sample Programs

This chapter describes how to build Oracle database applications with Pro*C/C++ using the sample programs that are included with this release.

This chapter contains these topics:

Sample Program Descriptions

When you install Pro*C/C++, Oracle Universal Installer copies a set of Pro*C/C++ sample programs to the *ORACLE_BASE**ORACLE_HOME*\precomp\demo\proc directory. These sample programs are listed in Table 3–1, "Sample Programs" and described in subsequent section.

When built, the sample programs that Oracle provides produce .exe executables.

For some sample programs, as indicated in the Notes column of the table, you must run the SQL scripts in the sample directory before you precompile and run the sample program. The SQL scripts set up the correct tables and data so that the sample programs run correctly. These SQL scripts are located in the *ORACLE_BASE**ORACLE_HOME*\precomp\demo\sql directory.

Oracle recommends that you build and run these sample programs to verify that Pro*C/C++ has been installed successfully and operates correctly. You can delete the programs after you use them.

You can build the sample program using a batch file called pcmake.bat or using Visual C++ 6.0. See "Building the Sample Programs" on page 3-9 for further information.

Table 3–1 Sample Programs

Sample Program	Source Files	Pro*C/C++ GUI Project File	MSVC Compiler Project File	Notes
ANSIDYN1	ansidyn1.pc	ansidyn1.pre	ansidyn1.dsp	
ANSIDYN2	ansidyn2.pc	ansidyn2.pre	ansidyn2.dsp	
COLDEMO1	coldemo1.h coldemo1.pc coldemo1.sql coldemo1.typ	coldemo1.pre	coldemo1.dsp	Run coldemo1.sql and the Object Type Translator before building coldemo1.
CPPDEMO1	cppdemo1.pc	cppdemo1.pre	cppdemo1.dsp	
CPPDEMO2	cppdemo2.pc empclass.pc cppdemo2.sql empclass.h	cppdemo2.pre	cppdemo2.dsp	Run cppdemo2.sql before building cppdemo2.
CPPDEMO3	cppdemo3.pc	cppdemo3.pre	cppdemo3.dsp	
CVDEMO	cv_demo.pc cv_demo.sql	cv_demo.pre	cv_demo.dsp	Run cv_demo.sql before building cv_demo.

Table 3–1 Sample Programs

Sample Program	Source Files	Pro*C/C++ GUI Project File	MSVC Compiler Project File	Notes
EMPCLASS	cppdemo2.pc empclass.pc cppdemo2.sql empclass.h	empclass.pre	empclass.dsp	Run cppdemo2.sql before building empclass.
LOBDEMO1	lobdemo1.h lobdemo1.pc lobdemo1.sql	lobdemo1.pre	lobdemo1.dsp	Run lobdemo1.sql before building lobdemo1.
MLTTHRD1	mltthrd1.pc mltthrd1.sql	mltthrd1.pre	mltthrd1.dsp	Run mltthrd1.sql before building mltthrd1.
NAVDEMO1	navdemo1.h navdemo1.pc navdemo1.sql navdemo1.typ	navdemo1.pre	navdemo1.dsp	Run navdemo1.sql and the Object Type Translator before building navdemo1.
OBJDEMO1	objdemo1.h objdemo1.pc objdemo1.sql objdemo1.typ	objdemo1.pre	objdemo1.dsp	Run objdemo1.sql and the Object Type Translator before building objdemo1.
ORACA	oraca.pc oracatst.sql	oraca.pre	oraca.dsp	Run oracatst.sql before building oraca.
PLSSAM	plssam.pc	plssam.pre	plssam.dsp	
SAMPLE	sample.pc	sample.pre	sample.dsp	
SAMPLE1	sample1.pc	sample1.pre	sample1.dsp	
SAMPLE2	sample2.pc	sample2.pre	sample2.dsp	
SAMPLE3	sample3.pc	sample3.pre	sample3.dsp	
SAMPLE4	sample4.pc	sample4.pre	sample4.dsp	
SAMPLE5	sample5.pc exampbld.sql examplod.sql	sample5.pre	sample5.dsp	Run exampbld.sql, then run examplod.sql, before building sample5.
SAMPLE6	sample6.pc	sample6.pre	sample6.dsp	
SAMPLE7	sample7.pc	sample7.pre	sample7.dsp	
SAMPLE8	sample8.pc	sample8.pre	sample8.dsp	
SAMPLE9	sample9.pc calldemo.sql	sample9.pre	sample9.dsp	Run calldemo.sql before building sample9.

Table 3–1 Sample Programs

Sample Program	Source Files	Pro*C/C++ GUI Project File	MSVC Compiler Project File	Notes
SAMPLE10	sample10.pc	sample10.pre	sample10.dsp	
SAMPLE11	sample11.pc sample11.sql	sample11.pre	sample11.dsp	Run sample11.sql before building sample11.
SAMPLE12	sample12.pc	sample12.pre	sample12.dsp	
SQLVCP	sqlvcp.pc	sqlvcp.pre	sqlvcp.dsp	
WINSAM	resource.h winsam.h winsam.ico winsam.pc winsam.rc	winsam.pre	winsam.dsp	

The following subsections describe the functionality of the sample programs.

ANSIDYN1
Demonstrates using ANSI dynamic SQL to process SQL statements that are not known until runtime. This program is intended to demonstrate the simplest (though not the most efficient) approach to using ANSI dynamic SQL.

ANSIDYN2
Demonstrates using ANSI dynamic SQL to process SQL statements that are not known until runtime. This program uses the Oracle extensions for batch processing and reference semantics.

COLDEMO1
Fetches census information for California counties. This program demonstrates various ways to navigate through collection-typed database columns.

CPPDEMO1
Prompts the user for an employee number, then queries the emp table for the employee's name, salary, and commission. This program uses indicator variables (in an indicator struct) to determine whether the commission is NULL.

CPPDEMO2
Retrieves the names of all employees in a given department from the emp table (dynamic SQL Method 3).

CPPDEMO3
Finds all salespeople and prints their names and total earnings (including commissions). This program is an example of C++ inheritance.

CVDEMO
Declares and opens a ref cursor.

EMPCLASS
The `EMPCLASS` and `CPPDEMO2` files were written to provide an example of how to write Pro*C/C++ programs within a C++ framework. `EMPCLASS` encapsulates a specific query on the `emp` table and is implemented using a cursor variable. `EMPCLASS` instantiates an instance of that query and provides cursor variable functionality (that is: `open`, `fetch`, `close`) through C++ member functions that belong to the emp class. The `empclass.pc` file is NOT a standalone demo program. It was written to be used by the `cppdemo2` demo program. To use the emp class, you have to write a driver (`cppdemo2.pc`) which declares an instance of the emp class and issues calls to the member functions of that class.

LOBDEMO1
Fetches and adds crime records to the database based on the person's Social Security number. This program demonstrates the mechanisms for accessing and storing large objects (LOBs) to tables and manipulating LOBs through the stored procedures available through the `DBMS_LOB` package.

MLTTHRD1
Shows how to use threading in conjunction with precompilers. The program creates as many sessions as there are threads. See "Multithreaded Applications" on page 2-15.

NAVDEMO1
Demonstrates navigational access to objects in the object cache.

OBJDEMO1
Demonstrates the use of objects. This program manipulates the object types *person* and *address*.

ORACA
Demonstrates how to use ORACA to determine various performance parameters at runtime.

PLSSAM
Demonstrates the use of embedded PL/SQL blocks. This program prompts you for an employee name that already resides in a database. It then executes a PL/SQL block, which returns the results of four `SELECT` statements.

SAMPLE
Adds new employee records to the personnel database and checks database integrity. The employee numbers in the database are automatically selected using the current maximum employee number +10.

SAMPLE1
Logs on to an Oracle database, prompts the user for an employee number, queries the database for the employee's name, salary, and commission, and displays the result. The program continues until the user enters 0 as the employee number.

SAMPLE2
Logs on to an Oracle database, declares and opens a cursor, fetches the names, salaries, and commissions of all salespeople, displays the results, and closes the cursor.

SAMPLE3
Logs on to an Oracle database, declares and opens a cursor, fetches in batches using arrays, and prints the results using the `print_rows()` function.

SAMPLE4
Demonstrates the use of type equivalencing using the `LONG VARRAW` external datatype.

SAMPLE5
Prompts the user for an account number and a debit amount. The program verifies that the account number is valid and that there are sufficient funds to cover the withdrawal before it debits the account. This program shows the use of embedded SQL.

SAMPLE6
Creates a table, inserts a row, commits the insert, and drops the table (dynamic SQL Method 1).

SAMPLE7
Inserts two rows into the `emp` table and deletes them (dynamic SQL Method 2).

SAMPLE8
Retrieves the names of all employees in a given department from the emp table (dynamic SQL Method 3).

SAMPLE9
Connects to an Oracle database using the scott/tiger account. The program declares several host arrays and calls a PL/SQL stored procedure (GET_EMPLOYEES in the CALLDEMO package). The PL/SQL procedure returns up to ASIZE values. The program keeps calling GET_EMPLOYEES, getting ASIZE arrays each time, and printing the values, until all rows have been retrieved.

SAMPLE10
Connects to an Oracle database using your username and password and prompts for a SQL statement. You can enter any legal SQL statement, but you must use regular SQL syntax, not embedded SQL. Your statement is processed. If the statement is a query, the rows fetched are displayed (dynamic SQL Method 4).

SAMPLE11
Fetches from the emp table, using a cursor variable. The cursor is opened in the stored PL/SQL procedure open_cur, in the EMP_DEMO_PKG package.

SAMPLE12
Demonstrates how to do array fetches using dynamic SQL Method 4.

SQLVCP
Demonstrates how you can use the sqlvcp() function to determine the actual size of a VARCHAR struct. The size is then used as an offset to increment a pointer that steps through an array of VARCHARs.

This program also demonstrates how to use the SQLStmtGetText() function to retrieve the text of the last SQL statement that was executed.

WINSAM
Adds new employee records to the personnel database and checks database integrity. You can enter as many employee names as you want and perform the SQL commands by selecting the appropriate buttons in the Employee Record dialog box. This is a GUI version of the sample program.

Building the Demonstration Tables

To run the sample programs, you must have a database account with the username `scott` and the password `tiger`. Also, you must have a database with the sample tables `emp` and `dept`. This account is included in the starter database for your Oracle9*i* server. If the account does not exist on your database, create the account before running the sample programs.

See *Oracle9i Database Administrator's Guide for Windows* for more information. If your database does not contain `emp` and `dept` tables, you can use the `demobld.sql` script to create them.

To build the sample tables:

1. Start SQL*Plus
2. Connect as username `scott` with the password `tiger`.
3. Run the `demobld.sql` script:

```
SQL> @ORACLE_BASE\ORACLE_HOME\sqlplus\demo\demobld.sql;
```

Building the Sample Programs

You can build the sample programs two ways:

- Using the `pcmake.bat` file provided
- Using Microsoft Visual C++ 6.0

Using pcmake.bat

The `pcmake.bat` file for compiling Pro*C/C++ demos is found in the following location:

```
ORACLE_BASE\ORACLE_HOME\precomp\demo\proc
```

This batch file is designed to illustrate how Pro*C/C++ applications can be built at the command prompt.

In order to use this batch file, Microsoft Visual Studio must be installed. The environment variable `MSVCDir` must be set. Pro*C/C++ command line options and linker options vary depending on your application.

You can use this file to build a demo, to build `sample1` for example:

1. Navigate to the location of the demo file and enter the following at the command prompt:

   ```
   C:\> CD ORACLE_BASE\ORACLE_HOME\precomp\demo\proc\sample1
   ```

2. Enter the following:

   ```
   % pcmake sample1
   ```

Using Microsoft Visual C++

Microsoft Visual C++ 6.0 project files have an extension of `.dsp`. The `.dsp` files in the *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\precomp\demo\proc` directory guide and control the steps necessary to precompile, compile, and link the sample programs.

Pro*C/C++, SQL*Plus, and the Object Type Translator have been integrated into the Microsoft Visual C++ sample project files. You do not have to run Pro*C/C++, SQL*Plus, and the Object Type Translator separately before compilation. See Appendix A, "Integrating Pro*C/C++ into Microsoft Visual C++" for more information.

> **See Also:** For more information on OTT, see the *Pro*C/C++ Precompiler Programmer's Guide.*

To build a sample program:

1. Open a Visual C++ project file, such as `sample1.dsp`.
2. Check the paths in the project file to ensure that they correspond to the configuration of your system. If they do not, change the paths accordingly. Your system may produce error messages if the paths to all components are not correct. See "Setting the Precompiler Options" on page 2-8 and "Setting the Path for the Sample .pre Files" on page 3-10.

   Note: All of the sample programs were created with `C:\oracle\ora90` as the default drive.

3. Choose Build > Rebuild All. Visual C++ creates the executable.

Setting the Path for the Sample .pre Files

By default the sample `.pre` files search for their corresponding `.pc` files in the `C:\oracle\ora90` directory where `C:\` is the drive that you are using, and `oracle\ora90` represents the location of the Oracle home. If the Oracle base and Oracle home directories are different on your computer, you must change the directory path to the correct path.

To change the directory path for a sample `.pre` file:

1. In Pro*C/C++, open the `.pre` file.
2. Double-click the filename in the Input File area to display the Input File dialog box.
3. Change the directory path to the correct path.
4. Click Open.

A

Integrating Pro*C/C++ into Microsoft Visual C++

This appendix describes how to integrate Pro*C/C++ into the Microsoft Visual C++ integrated development environment.

This appendix contains these topics:

- Integrating Pro*C/C++ within Microsoft Visual C++ Projects
- Adding Pro*C/C++ to the Tools Menu

Integrating Pro*C/C++ within Microsoft Visual C++ Projects

This section describes how to fully integrate Pro*C/C++ within Microsoft Visual C++ projects.

All the precompiler errors and warnings are displayed in the output box where Microsoft Visual C++ displays compiler and linker messages. You do not have to precompile a file separately from the Microsoft Visual C++ build environment. More importantly, Microsoft Visual C++ maintains the dependencies between `.c` and `.pc` files. Microsoft Visual C++ maintains the dependency and precompile files, if needed.

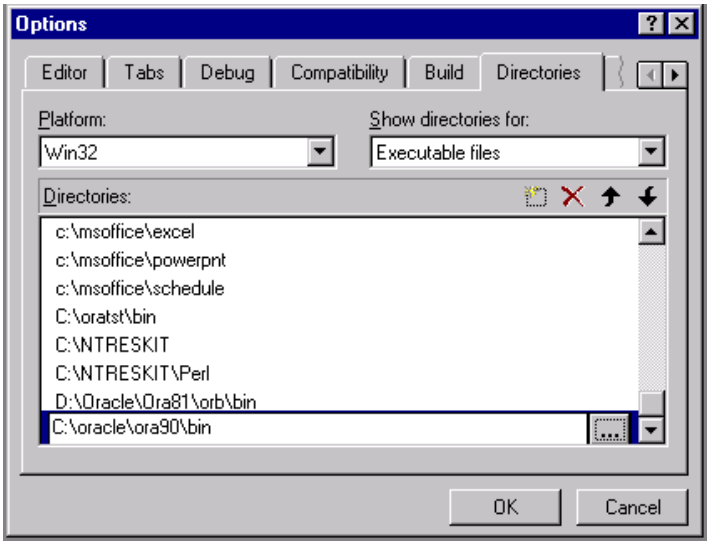
All of the procedures in this section are performed within Microsoft Visual C++.

Specifying the Location of the Pro*C/C++ Executable

For Microsoft Visual C++ to run Pro*C/C++, it must to know the location of the Pro*C/C++ executable. If Microsoft Visual C++ was installed before any Oracle Release 9.0.1 products were installed, then you must add the directory path.

To specify the location of the Pro*C/C++ executable:

1. Choose Options from the Tools menu. The Options dialog box appears.



2. Click the Directories tab.

3. Select Executable files from the Show directories for list box.
4. Scroll to the bottom of the Directories box and click the dotted rectangle.
5. Enter the *ORACLE_BASE**ORACLE_HOME*\bin directory. For example:

   ```
   C:\oracle\ora90\bin
   ```

6. Click OK.

Specifying the Location of the Pro*C/C++ Header Files

To specify the location of the Pro*C/C++ header files:

1. Choose Options from the Tools menu. The Options dialog box appears.
2. Click the Directories tab.
3. Select Include Files from the Show directories for list box.
4. Scroll to the bottom of the Directories box and click the dotted rectangle.
5. Enter the *ORACLE_BASE**ORACLE_HOME*\precomp\public directory. For example:

   ```
   C:\oracle\ora90\precomp\public
   ```

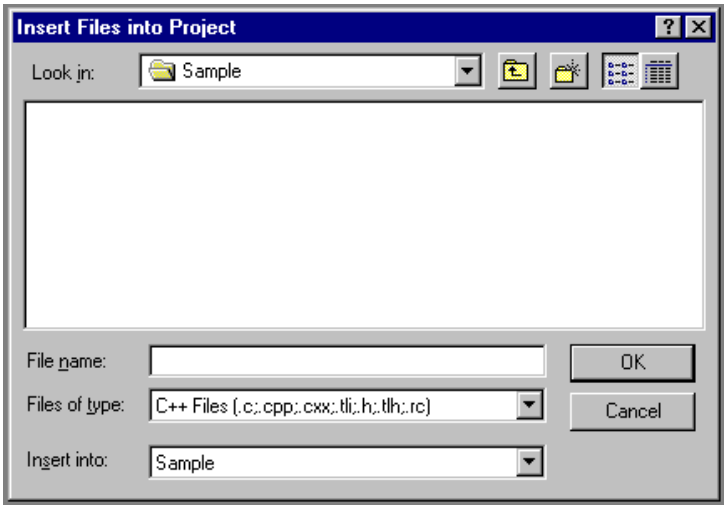
6. Click OK.

Adding .pc Files to a Project

After you create a project, you need to add the .pc file(s).

To add a .pc file to a project:

1. Choose Add To Project from the Project menu, and then choose Files. The Insert Files into Project dialog box appears.



2. Select All Files from the Files of type list box.
3. Select the .pc file.
4. Click OK.

Adding References to .c Files to a Project

For each .PC file, you need to add a reference to the .C file that will result from precompiling.

To add a reference to a .c file to a project:

1. Choose Add To Project from the Project menu, and then choose Files. The Insert Files into Project dialog box appears.
2. Type the name of the .c file in the File Name box.

3. Click OK. Because the `.c` file has not been created yet, Microsoft Visual C++ displays the following message: "The specified file does not exist. Do you want to add a reference to the project anyway?"
4. Click Yes.

Adding the Pro*C/C++ Library to a Project

Pro*C/C++ applications must link with the library file `orasql9.lib`.

To add the Pro*C/C++ library to a project:

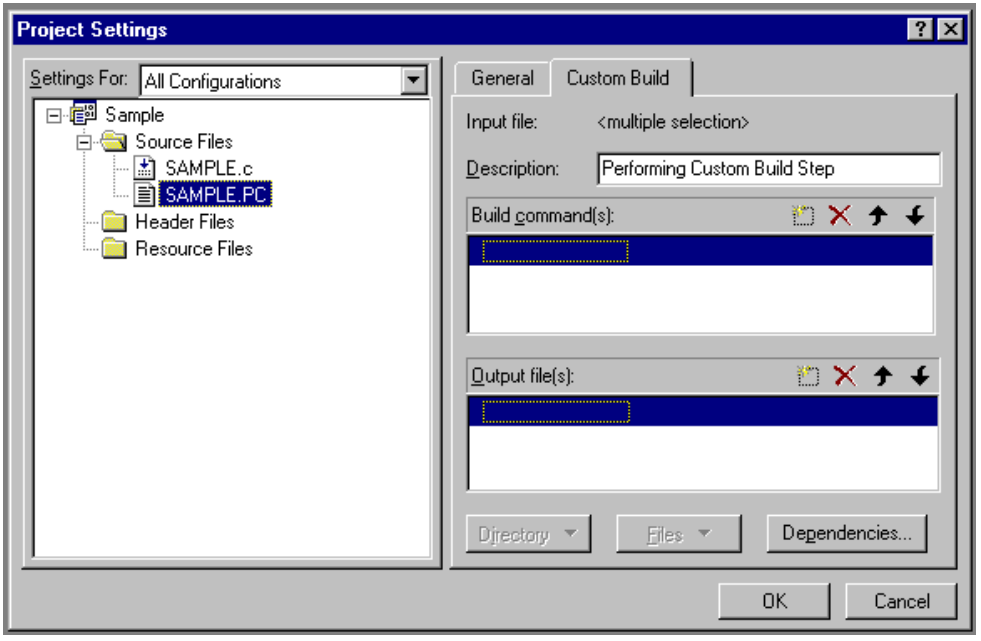
1. Choose Add To Project from the Project menu, and then choose Files.

   The Insert Files into Project dialog box appears.
2. Select All Files from the Files of type list box.
3. Select `orasql9.lib` from the *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\precomp\lib\msvc` directory.
4. Click OK.

Specifying Custom Build Options

To specify custom build options:

1. In FileView, right-click a `.pc` file and choose Settings. The Project Settings dialog box appears with the Custom Build tab displayed.



2. In the Build command(s) box, on one line, set the build to use the same hardcoded path as that of the `$ORACLE_HOME` setting.

3. In the Output file(s) box, enter one of the following:

If You Are Generating...	Enter...
.C files	`$(ProjDir)\$(InputName).c`
.CPP files	`$(ProjDir)\$(InputName).cpp`

`$(ProjDir)` and `$MSDEVDIR` are macros for custom build commands in Microsoft Visual C++. When the project is built, Microsoft Visual C++ will check the date of the output files to determine whether they need to be rebuilt for any

new modifications made to the source code. See the Microsoft Visual C++ documentation for more information.

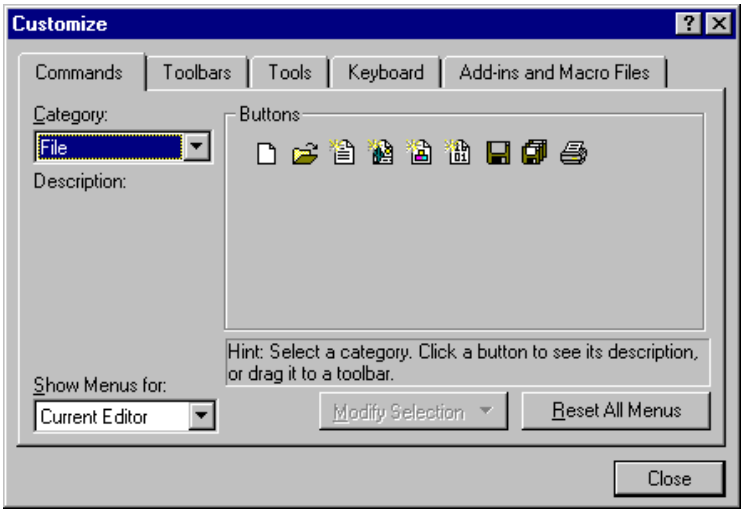
4. Click OK.

Adding Pro*C/C++ to the Tools Menu

You can include Pro*C/C++ as a choice in the Tools menu of Microsoft Visual C++.

To add Pro*C/C++ to the Tools menu:

1. From within Microsoft Visual C++, choose Customize from the Tools menu. The Customize dialog box appears.



2. Click the Tools tab.
3. Scroll to the bottom of the Menu contents box and click the dotted rectangle.
4. Enter the following text:

   ```
   Pro*C/C++
   ```

5. In the Command box, type the path and filename of the graphical Pro*C/C++ executable, or use the Browse button to the right of the box to select the file name. For example:

   ```
   C:\oracle\ora90\bin\procui.exe
   ```

6. In the Arguments box, enter the following text.

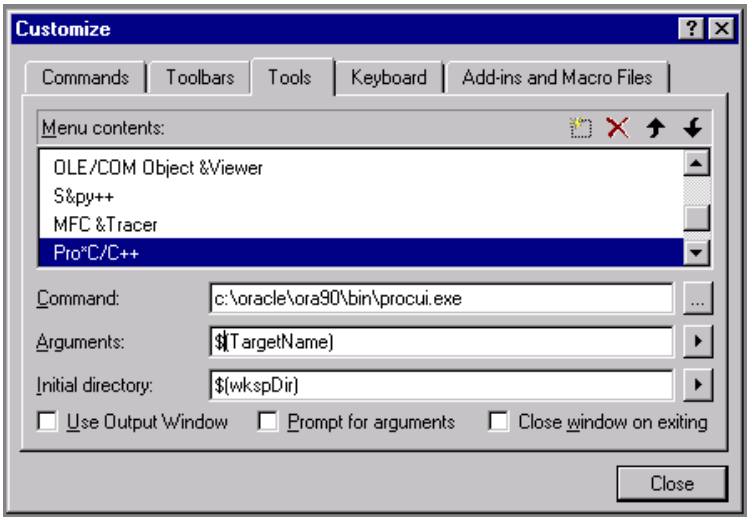
   ```
   $(TargetName)
   ```

   When you choose Pro*C/C++ from the Tools menu, Microsoft Visual C++ uses the $(TargetName) argument to pass the name of the current development project to Pro*C/C++. Pro*C/C++ then opens a precompile project with the same name as the opened project, but with a .PRE extension in the project directory.

7. In the Initial directory box, enter the following text:

   ```
   $(WkspDir)
   ```

   The Customize dialog box should now look like the following graphic (although the Oracle home directory may be different on your computer).



8. Click Close. Microsoft Visual C++ adds Pro*C/C++ to the Tools menu.

Index

Numerics

A

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T