ProC/C++^{TM} Precompiler

Getting Started

Release 8.0.6 for Windows NT and Windows 95/98

June 1999

Part No. A69520-01

ORACLE

Pro*C/C++ Precompiler Getting Started, Release 8.0.6 for Windows NT and Windows 95/98

Part No. A69520-01

Contributors: Riaz Ahmed, Eric Belden, Lisa Giambruno, Neeraj Gupta, Helen Slattery, Jeff Stein

Contents

Contact Us!

Oracle Pro*C/C++ Precompiler Getting Started, Release 8.0.6 for Windows NT and 95/98

Part No. A69520-01

This document describes how to contact Oracle Corporation if you have issues with the documentation or software.

Read the section...	If you...
How to Contact Oracle Technical Publications	Have issues with Documentation
How to Contact Oracle Support Services	Have issues with Software
Resources for Oracle Partners and Developers	Want to join an Oracle partner or application developer program

How to Contact Oracle Technical Publications

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. 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 about this manual?
- Do you have suggestions for improvement? Please indicate the chapter, section, and page number (if available).

You can send comments regarding documentation to *ntdoc@us.oracle.com*

How to Contact Oracle Support Services

Please copy this page and distribute within your organization as necessary.

Oracle Support Services can be reached at the following telephone numbers. The hours of business are detailed in your support contract and the *Oracle Customer Support Guide* in your kit.

Oracle Support Services In...	Call...
All locations	The telephone number for your country is listed at the following Web site: `http://www.oracle.com/support/contact_us/sup_hot_phone.html` Oracle Support Services telephone numbers are also listed in the *Oracle Customer Support Guide* in your kit.

Please complete the following checklist before you call. If you have this information ready, your call can be processed much quicker.

❏ Your CPU Support Identification Number (CSI Number) if applicable.

____

____

❏ The hardware name on which your application is running.

____

____

____

❏ The operating system name and release number on which your application is running.

____

____

____

- ❑ The release numbers of the Oracle Server and associated products involved in the current problem. For example, Oracle8 Server release 8.0.6.0.0 and Oracle Replication Manager release 1.6.0.0.0.

- ❑ Are you using 16-bit or 32-bit Oracle software?

- ❑ The third-party vendor and version you are using.

- ❑ The exact error codes and messages. Please write these down as they occur. They are critical in helping Oracle Support Services to quickly resolve your problem.

- ❑ A description of the issue, including:
  - **What happened?** For example, the command used and its result.

- **When did it happen?** For example, during peak system load, or after a certain command, or after an operating system upgrade.

  ____

  ____

  ____

- **Where did it happen?** For example, on a particular system, or within a certain procedure or table.

  ____

  ____

  ____

- **What is the extent of the problem?** For example, production system unavailable, or moderate impact but increasing with time, or minimal impact and stable.

  ____

  ____

  ____

❏ Keep copies of any trace files, core dumps, and redo log files recorded at or near the time of the incident. Oracle Support Services may need these to further investigate your problem.

Resources for Oracle Partners and Developers

This section provides information on partner programs and resources for Oracle database administrators and application developers.

Information Source	Description
Oracle Corporation Home Page `http://www.oracle.com`	This Web site is the starting point for general information on Oracle Corporation.
Alliance Online `http://alliance.oracle.com`	Oracle provides leading-edge technology, education, and technical support that enables you to effectively integrate Oracle into your business. By joining the Oracle Partner Program, you demonstrate to customers that you are committed to delivering innovative Oracle-based solutions and services. The greater your commitment to Oracle, the more we can help you grow your business. It's that simple. The value you derive is associated directly with your level of commitment.
Oracle Education `http://education.oracle.com/`	Customers come to Oracle Education with a variety of needs. You may require a complete curriculum based on your job role to enable you to implement new technology. Or you may seek an understanding of technology related to your key area of responsibility to help you meet technical challenges. You may be looking for self-paced training that can be used as an ongoing resource for reference and hands-on practice. Or, you may be interested in an overview of a new product upgrade. Whatever your training need, Oracle Education has the solution.
Oracle Technology Network `http://technet.oracle.com/`	The Oracle Technology Network is your definitive source for Oracle technical information for developing for the Internet platform. You will be part of an online community with access to free software, Oracle Technology Network-sponsored Internet developer conferences, and discussion groups on up-to-date Oracle technology. Membership is free.
Oracle Store `http://oraclestore.oracle.com/`	This is Oracle's online shopping center. Come to this site to find special deals on Oracle software, documentation, publications, computer-based training products, and much more.

Information Source	Description
Oracle Support Services' Support Web Center `http://www.oracle.com/support/`	Oracle Support Services offers a range of programs so you can select the support services you need and access them in the way you prefer: by telephone, electronically, or face to face. These award-winning programs help you maintain your investment in Oracle technology and expertise.
Oracle*MetaLink* `http://www.oracle.com/support/elec_sup/index.html`	Oracle*MetaLink* is Oracle Support Services' premier Web support service. It is available to Oracle*metals* customers (Gold, Silver, Bronze), 24 hours a day, seven days a week.
Oracle*Lifecycle* `http://www.oracle.com/support/sup_serv/lifecycle/index.html`	Oracle*Lifecycle* is designed to deliver customized, industry-focused, full life-cycle support solutions that enable industry leaders to use Oracle technology to make smart business decisions, achieve operational excellence, and succeed in their markets.
Expert*ONLINE* `http://www.oracle.com/support/sup_serv/online/index.html`	Oracle Support Services has launched a new line of services called Expert*ONLINE*. These services provide online database administration for companies looking to supplement their existing DBA staff or fill a DBA role. Services range from Expert*DETECT*, a monitoring, diagnostic, and recommendation service, to Expert*DBA*, a full online database administration service.
Virtual Support Analyst (VSA) `http://www.oracle.com/support/sup_serv/vsa_start.html`	VSA is Oracle's Internet e-mail service; it is available to U.S. customers with an Oracle*metals* support agreement. With VSA, you can initiate a request for assistance through e-mail, bypassing the queues you may encounter when using telephone support. VSA also enables you to access Oracle's bug database.
Customer Service `http://www.oracle.com/support/cus_serv/index.html`	This site provides resources to make your interactions with Oracle as easy as possible. Among the things you can do are ▪ Learn what is a CPU Support Identification (CSI) number ▪ Update your technical contact information ▪ Find out whom to contact for invoice and collection issues ▪ Request product update shipments ▪ Access a glossary of Oracle Support Services terms

Information Source	**Description**
U.S. Customer Visit Program `http://www.oracle.com/support/cus_serv/cus_visit.html`	This U.S.-based program has been established to help our customers understand and obtain maximum benefit from the support services they have purchased. The visit typically offers a customized orientation presentation, a comprehensive overview and demonstration of Oracle's electronic services, and helpful tips on working more effectively with Oracle Support Services.
Support Web Center Library `http://www.oracle.com/support/library/index.html`	This site contains articles, guides, and other documentation to help you leverage the wealth of knowledge and reference material that Oracle Support Services produces.
Product Availability *infowin@us.oracle.com*	Send an e-mail to request information on future product releases on Oracle for Windows NT and Windows 95/98.

Preface

This guide provides introductory information for the Oracle Pro*C/C++ precompiler running under the Microsoft Windows NT and Windows 95/98 operating systems. Specific topics discussed in this preface are:

- Prerequisites
- Intended Audience
- How This Guide Is Organized
- Conventions
- Documentation Library
- Related Documents

Prerequisites

This guide assumes that you are familiar with:

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

Intended Audience

This guide is necessary for anyone who wants to use the Oracle Pro*C/C++ precompiler under the Microsoft Windows NT or Windows 95/98 operating system.

How This Guide Is Organized

This guide is organized as follows:

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

Describes Pro*C/C++, the Oracle programmatic interface for the C and C++ languages running under the 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, "Building Pro*C/C++ Applications"

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.

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

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

Chapter B, "Integrating Pro*C/C++ into Borland C++"

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

Conventions

The following conventions are used in this guide.

Convention	Example	Meaning
All uppercase plain	ORANT\DATABASE\INITORCL.ORA	Indicates command names, SQL reserved words, and keywords, as in ALTER DATABASE. All uppercase plain is also used for directory names and file names.
Italic	Italic used to indicate a variable: *ORACLE_HOME*\filename Italic used for emphasis: The WHERE clause may be used to *join* rows in different tables.	Indicates a value that you must provide. For example, if a command asks you to type *filename*, you must type the actual name of the file. Italic is also used for emphasis in the text and to indicate the titles of other guides.
Oracle8 database		The database component of Oracle8.
C:\>	C:\> ORANT\DATABASE	Represents the Windows NT or Windows 95/98 command prompt of the current hard disk drive. Your prompt may differ and may, at times, reflect the subdirectory in which you are working. Referred to as the *MS-DOS command prompt* in this guide.
Backslash (\) before a directory name	\DATABASE	Indicates that the directory is a subdirectory of the root directory.
Oracle home	Go to the *ORACLE_HOME*\DATABASE directory.	Oracle home is represented as the hard drive letter and the top level directory where your Oracle software is installed. In this guide, the convention *ORACLE_HOME* is used to indicate your Oracle home directory, which may be: C:\ORANT for Windows NT; C:\ORAWIN95 for Windows 95; C:\ORAWIN98 for Windows 98; or whatever you may have called your Oracle home.

Convention	Example	Meaning
HOME_NAME	Oracle*HOME_NAME*TNSListener	Represents the Oracle home name if you use multiple Oracle homes. This convention is not applicable for a single Oracle home. The home name can be up to sixteen alphanumeric characters. The only special character allowed in the home name is the underscore.
Symbols	period . comma , hyphen - semicolon ; colon : equal sign = backslash \\ single quote ‘ double quote “ parentheses ()	Symbols other than brackets and vertical bars must be entered in commands exactly as shown.

Documentation Library

This guide is part of a larger library of Oracle documentation. The Oracle documentation library consists of two types of documentation:

Documentation Type	Describes...
Operating System-specific	Installation, configuration, and use of Oracle products in a Windows NT or Windows 95/98 environment. Operating system-specific documents are occasionally referred to in the generic documentation set. These documents are easy to identify because they always mention their specific operating system in their title.

Documentation Type	Describes...
Generic	Oracle database, Oracle networking, and Application Programming Interfaces information that is uniform across all operating system platforms. The majority of documents in your documentation set belong to this category. While reading through the generic documentation set, you are occasionally asked to refer to your platform (or operating system) documentation for procedures specific to the Windows NT or Windows 95/98 operating systems. To easily identify where these generic documentation references are described in your operating system documentation, see the index of this guide for the following entry: generic documentation references All generic documentation references described in this guide appear under this index entry.

Related Documents

For more information, see the following manuals.

- *Oracle8 Enterprise Edition for Windows NT Installation CD-ROM Insert*
- *Oracle8 Enterprise Edition Release Notes*
- *Oracle8 Enterprise Edition Getting Started for Windows NT*
- *Oracle Enterprise Manager Configuration Guide*
- *Oracle Enterprise Manager Administrator's Guide*
- *Net8 Getting Started for Windows NT and Windows 95*
- *Net8 Administrator's Guide*
- *Oracle8 Parallel Server Concepts and Administration*
- *Oracle Parallel Server Management User's Guide*
- *Oracle8 Concepts*
- *Oracle8 Reference*
- *Oracle8 Error Messages*
- *Pro*C/C++ Precompiler Programmer's Guide*

1
Introducing Pro*C/C++

This chapter describes Pro*C/C++, the Oracle programmatic interface for the C and C++ languages running under the Window NT and Windows 95/98 operating systems. Pro*C/C++ enables you to build Oracle database applications in a Win32 environment. Specific topics discussed are:

- What is Pro*C/C++?
- PL/SQL Support
- Features

What is Pro*C/C++?

The Pro*C/C++ precompiler takes SQL statements embedded in your C and C++ program and converts them to standard C code. When you precompile this code, the result is a C or C++ program that you can compile and use to build applications that access an Oracle database.

To access an Oracle database, you use a high-level query language called Structured Query Language (SQL). Pro*C/C++ is a precompiler that converts SQL statements in EXEC SQL commands into C statements so that the resulting output file can then be compiled by a C/C++ compiler.

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

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

PL/SQL Support

Pro*C/C++ supports PL/SQL, Oracle's procedural language extension to the SQL language standard for enhancing performance of the Oracle database.

Additional Information: See the *PL/SQL User's Guide and Reference* for additional information on PL/SQL. See the *Pro*C/C++ Precompiler Programmer's Guide* for information on embedding PL/SQL in your Pro*C/C++ applications.

Features

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

- remote by way of Net8 or local access to Oracle8 Servers
- 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 8.0 and host language arrays in PL/SQL procedures
- multi-threaded applications
- full ANSI C compliance
- Microsoft Visual C++ support, version 5.0 and 6.0 for 32-bit applications
- Borland C++ support, version 5.0 for 32-bit applications

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. Specific topics discussed are:

Note: For additional information, see the Getting Started guide and the Release Notes that are provided with the product.

Oracle Directory Structure

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

The subdirectories in *ORACLE_HOME* contain the Pro*C/C++ executable files and library files.

In the following table, the letters *nn* represent the version of the product installed. For example, if you previously installed Pro*C/C++ version 2.2, you have a \PRO22 subdirectory under *ORACLE_HOME*. If you now install Pro*C/C++ version 8.0, Oracle Installer adds a \PRO80 subdirectory.

Directory Name	Contents
\RDBMS*nn*	Oracle8 server message, resource, README, and database files.
\BIN	Executable programs, batch files, and dynamic link libraries (DLLs).
\PRO*nn*	Subdirectories that contain files and subdirectories for Pro*C/C++.
\ORAINST	Oracle Installer utilities and other files.
\PLSQL*nn*	Message files, SQL scripts, and demonstration files for PL/SQL.
\DBS	Files for Toolkit II and message files for products created for Oracle Installer.
\OCI*nn*	Executable programs and files for the Oracle Call Interface.

Based on the supported products that you install, Oracle Installer also creates the appropriate product subdirectory. When you install Pro*C/C++ version 8.0, Oracle Installer creates the \PRO80 subdirectory, which contains the following subdirectories:

Directory Name	Contents
\C	Contains the INCLUDE and SAMPLES subdirectories.
\INCLUDE	Contains header files for Pro*C/C++.

Directory Name	Contents
\SAMPLES	Contains one subdirectory for each sample program.
\LIB	Contains the BORLAND and MSVC subdirectories.
\BORLAND	Contains Pro*C/C++ libraries that are compatible with Borland C++ 5.0.
\MSVC	Contains Pro*C/C++ libraries that are compatible with Microsoft Visual C++ 5.0 and 6.0.

The Pro*C/C++ installation procedure copies a set of 32-bit sample programs and their corresponding make files to the *ORACLE_HOME*\PRO80\C\SAMPLES directory. Samples can be found in their corresponding subdirectories. Oracle recommends that you build and run these sample programs to verify that Pro*C/C++ has been installed successfully and operates correctly. See "Building and Running Sample Programs" on page 3-2.

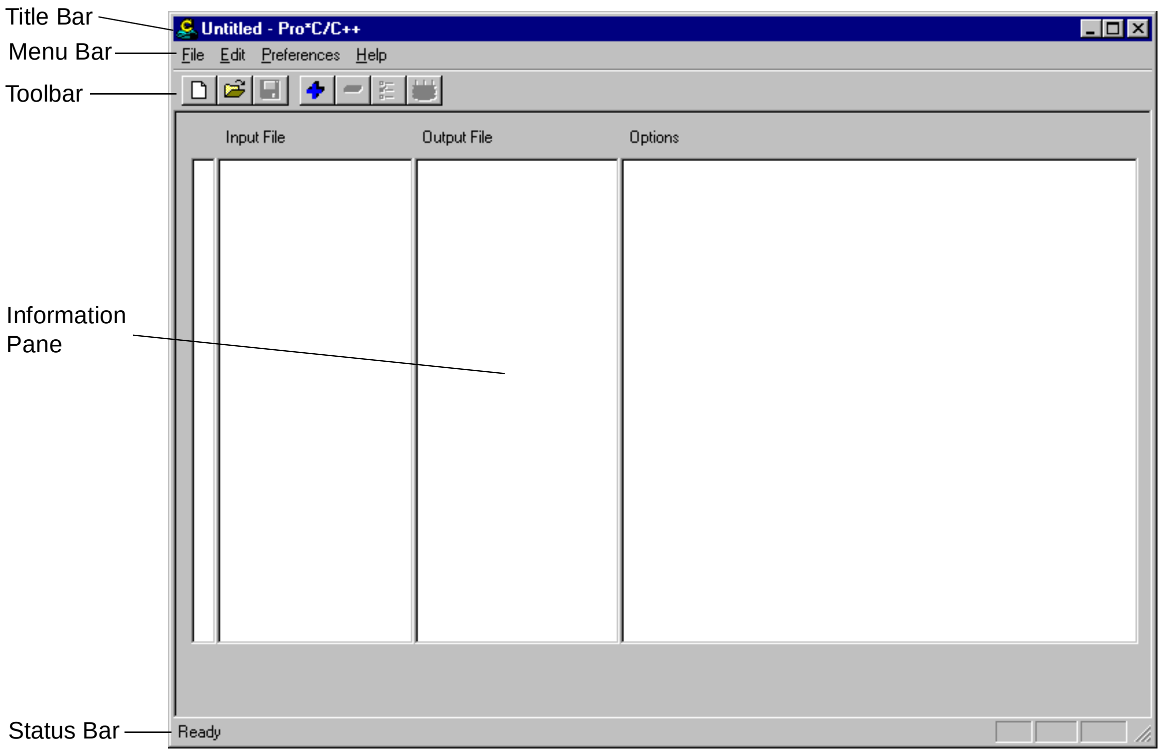
Using the Graphical User Interface

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

To start the graphical user interface, do one of the following:

- Choose Start > Programs > Oracle for Windows NT - [*HOME_NAME*] > Oracle Pro C_C++ 8.0
- Choose Start > Programs > Oracle for Windows 95/98 - [*HOME_NAME*] > Oracle Pro C_C++ 8.0
- Enter `procui80.exe` at the MS-DOS command prompt

The Pro*C/C++ precompile environment contains five elements:



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.

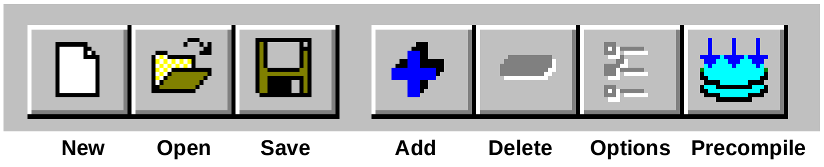
Menu Bar

Contains the following menus:

Menu	Description
File	Contains commands to create a project, open an existing project, save a project under the same name or under a different name, specify a connect string to an Oracle database, precompile a project, or exit the application.
Edit	Contains commands to add files to a project, delete files from a project, and change options for the selected file(s).
Preferences	Contains commands to choose the manner in which a name is assigned to the output files.
Help	Contains the About Pro*C/C++ command, which displays the version number of the application and copyright information.

Toolbar

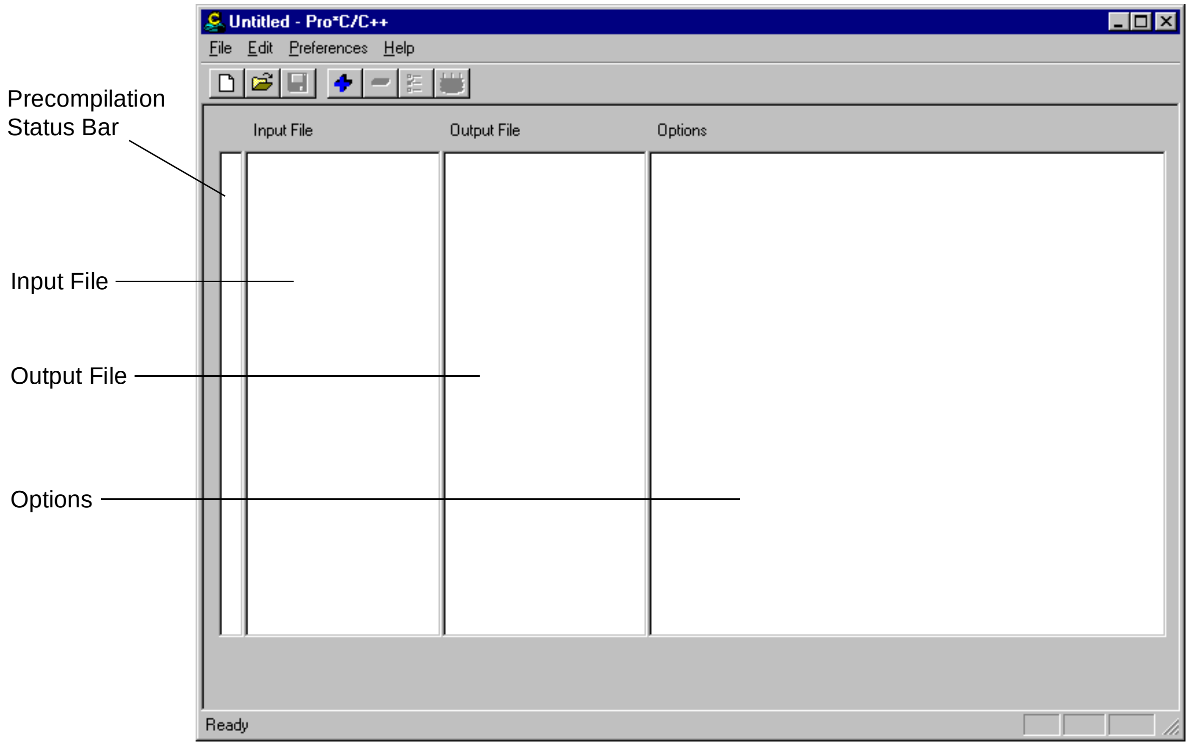
Enables you to execute commands by clicking a button:



Button	Description
New	Create a new project file.
Open	Open an existing project file.
Save	Save the active project file to disk.
Add	Add files to the precompile list.
Delete	Delete files from the precompile list.
Options	Display or change precompile options.
Precompile	Precompile the selected files.

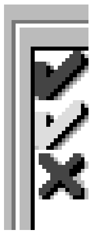
Information Pane

Consists of four elements:



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

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


green check--successful precompile
yellow check--successful precompile with warning(s)
red X--unsuccessful precompile

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

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.

First, start the Pro*C/C++ application by doing one of the following:

- Choose Start > Programs > Oracle for Windows NT - [*HOME_NAME*] > Oracle Pro C_C++ 8.0
- Choose Start > Programs > Oracle for Windows 95/98 - [*HOME_NAME*] > Oracle Pro C_C++ 8
- Enter `procui80.exe` at the MS-DOS command prompt

Then, perform the following steps:

- Step 1: Open a Project
- Step 2: Set the Default Output File Name (Optional)
- Step 3: Add Files to the Project
- Step 4: Delete Files from the Project (Optional)

- Step 5: Set the Precompile Options (Optional)
- Step 6: Specify Database Connection Information (Optional)
- Step 7: Precompile
- Step 8: Check the Results
- Step 9: Fix Any Errors

The remainder of this section describes each of these steps in detail.

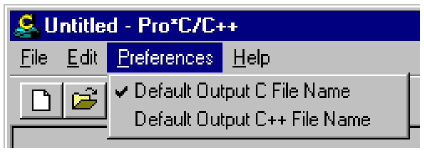
Step 1: Open a Project

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 New Project from the File menu.
- To open an existing project, choose Open Project from the File menu.

Step 2: Set the Default Output File Name (Optional)

Use the Preferences menu to choose the manner in which a name is assigned to the output files.



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

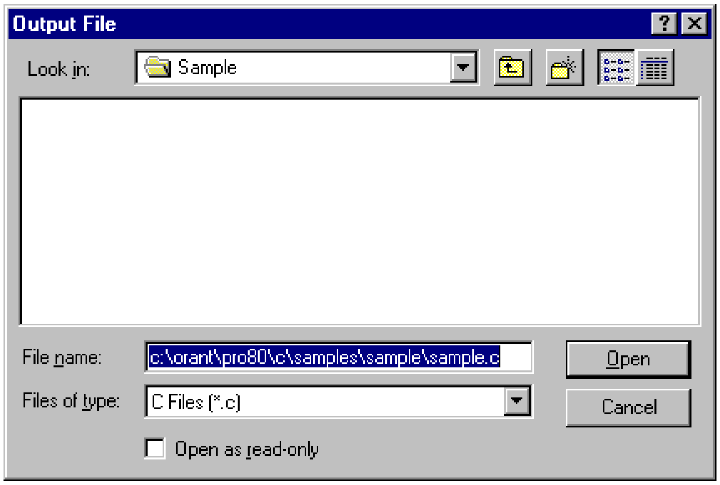
- If you select Default Output C File Name, the default extension of the output files is .C.
- If you select Default Output C++ File Name, the default extension of the output files is .CPP.
- If you deselect both Default Output C File Name and Default Output C++ File Name, the *Output File* dialog box appears when you add an output file.

  Enter an output file name for the file selected. Once you select or enter a file name, it appears in the Output File area of the information pane.

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

1. Double-click the file name in the Input File or Output File area of the information pane.

   The *Input File* or *Output File* dialog box appears.



File name: c:\orant\pro80\c\samples\sample\sample.c

Files of type: C Files (*.c)

Open as read-only

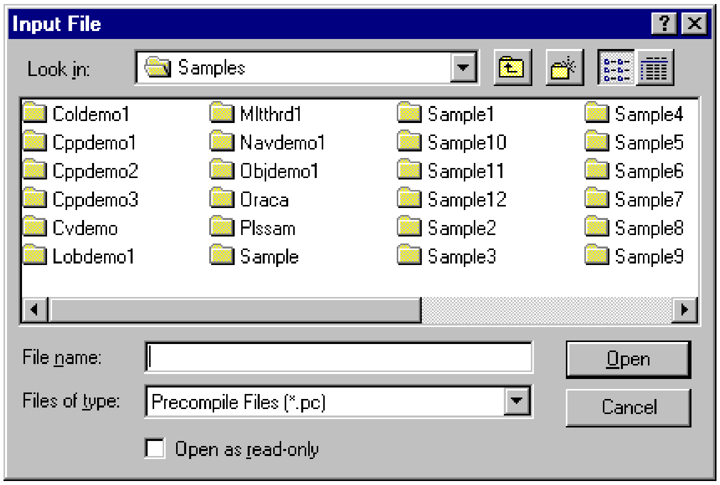
2. Replace the old file name with the new file name.
3. Click Open.

Step 3: Add Files to the Project

A Pro*C/C++ project consists of one or more precompilable files.

To add files to the project:

1. Choose Add from the Edit menu. The *Input File* dialog box appears.



2. Select one or more file names.

3. Click Open.

   The selected files appear in the information pane.

Step 4: Delete Files from the Project (Optional)

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 Delete from the Edit menu.

3. Click Yes.

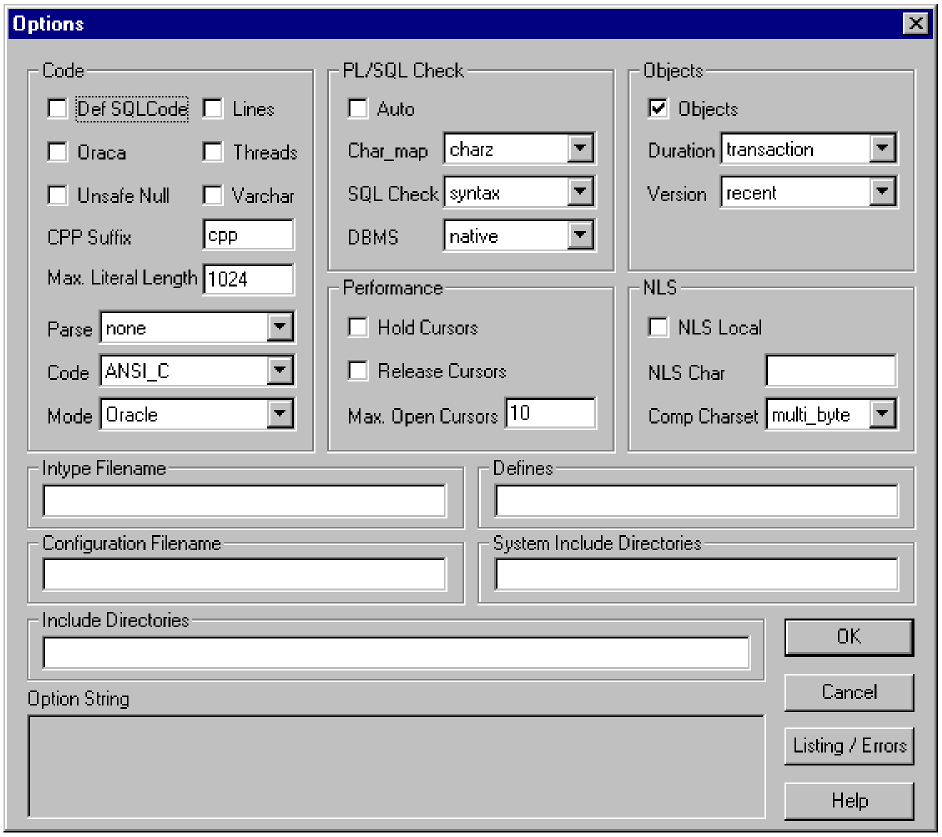
Step 5: Set the Precompile Options (Optional)

Many useful options are available. They 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. Highlight one or more files in the Input File list.
2. Choose Options from the Edit menu.

   The *Options* dialog box appears.


Options

Code
- [ ] Def SQLCode
- [ ] Lines
- [ ] Oraca
- [ ] Threads
- [ ] Unsafe Null
- [ ] Varchar
- CPP Suffix: cpp
- Max. Literal Length: 1024
- Parse: none
- Code: ANSI_C
- Mode: Oracle

PL/SQL Check
- [ ] Auto
- Char_map: charz
- SQL Check: syntax
- DBMS: native

Objects
- [✓] Objects
- Duration: transaction
- Version: recent

Performance
- [ ] Hold Cursors
- [ ] Release Cursors
- Max. Open Cursors: 10

NLS
- [ ] NLS Local
- NLS Char:
- Comp Charset: multi_byte

Intype Filename

Defines

Configuration Filename

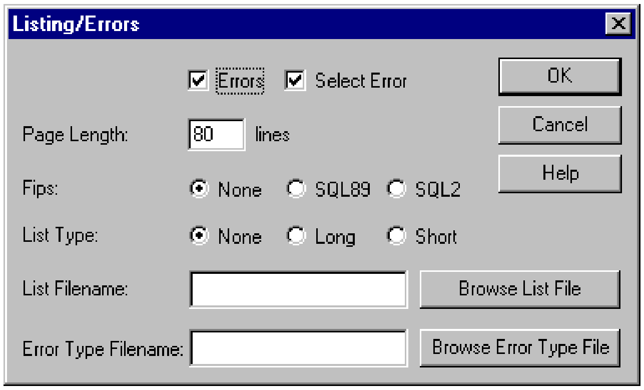
System Include Directories

Include Directories

Option String

3. 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.

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



[✓ Errors] [✓ Select Error]

Page Length: 80 lines

Fips: [✓ None] SQL89 SQL2

List Type: [✓ None] Long Short

List Filename:

Error Type Filename:

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

5. After you set the options in the *Options* dialog box, click OK.

Additional Information: See Chapter 9 in the *Pro*C/C++ Precompiler Programmer's Guide* for a description of the precompiler options. In addition, click the Help button in the *Options* dialog box for a condensed version of the information in Chapter 9.

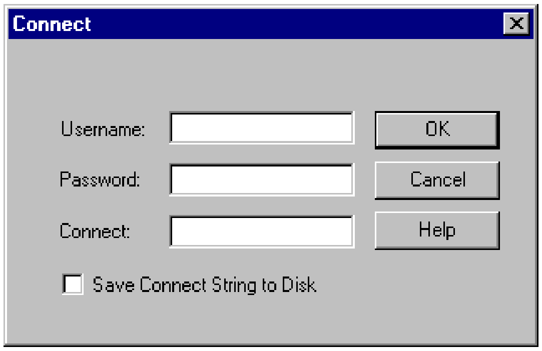
See "Using the Precompiler Options" on page 2-17 for issues that are related to Pro*C/C++ for Windows NT and Windows 95/98.

Step 6: Specify Database Connection Information (Optional)

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 Connect from the File menu. 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 SQLCHECKing.
3. The *Connect* dialog box appears automatically at precompile time if you have not previously responded. Enter the user name, the password, and the connect string (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. Click OK.

Step 7: Precompile

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

To precompile:

1. Highlight one or more files in the Input File list. You can use the Ctrl button to highlight files that are not adjacent to each other (for example, the first and third files in a list).
2. Choose Precompile from the File menu.

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

3. Click OK.

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

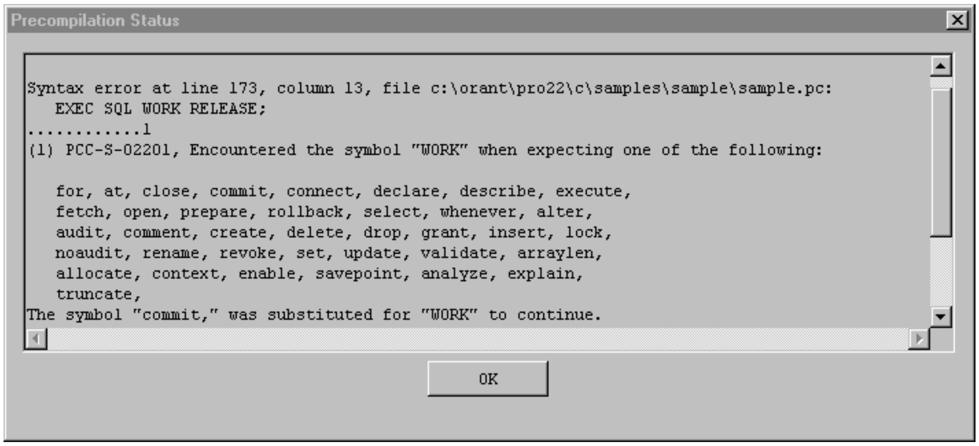
Step 8: Check the Results

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.

Step 9: Fix Any Errors

Double-click the yellow check or red X to display the *Precompilation Status* dialog box, which lists warning messages or reasons why the precompilation failed.



```
Syntax error at line 173, column 13, file c:\orant\pro22\c\samples\sample\sample.pc:
   EXEC SQL WORK RELEASE;
............1
(1) PCC-S-02201, Encountered the symbol "WORK" when expecting one of the following:

   for, at, close, commit, connect, declare, describe, execute,
   fetch, open, prepare, rollback, select, whenever, alter,
   audit, comment, create, delete, drop, grant, insert, lock,
   noaudit, rename, revoke, set, update, validate, arraylen,
   allocate, context, enable, savepoint, analyze, explain,
   truncate,
The symbol "commit," was substituted for "WORK" to continue.
```

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

To exit Pro*C/C++, choose Exit from the File menu. 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

If you perform most of your programming tasks at the command line, it is recommended that you run Pro*C/C++ at the command line.

To precompile an application with Pro*C/C++, enter the following command:

```
C:\> PROC INAME=filename.pc
```

where *filename*.pc is the name of the source file.

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

If the source file is not within your current working directory, include the file's full path after the INAME argument.

Header Files

The *ORACLE_HOME*\PRO80\C\INCLUDE directory contains the Pro*C/C++ header files.

Additional Information: See the *Pro*C/C++ Precompiler Programmer's Guide* for more information about ORACA.H, SQLCA.H, and SQLDA.H.

ORACA.H

Consists of an extended set of diagnostic tools for runtime errors and status changes. ORACA handles Oracle communications. Note that using the ORACA adds to runtime overhead.

SQL2OCI.H

Consists of 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

Consists of ANSI prototypes for externalized functions that can be used for backwards compatibility with V7 OCI.

SQLCA.H

Consists of the host language data structure. Oracle updates the SQLCA after every executable SQL statement (values are undefined after a declarative statement). By checking Oracle return codes stored in the SQLCA, your application can determine the outcome of a SQL statement.

SQLCPR.H

Provides 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

Consists of the host program data structure that contains descriptions of select-list items or input host variables. SQLDA files differ among host languages and operating systems.

SQLKPR.H

Consists of K&R prototypes for externalized functions that can be used for backwards compatibility with V7 OCI.

SQLPROTO.H

SQLPROTO.H 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_HOME*\PRO80\C\INCLUDE directory.

If You Receive a PCC-S-02014 Error...

If you receive a PCC-S-02014 error when precompiling, copy the batch files MOD_INCL.BAT and ADD_NEWL.BAT from the *ORACLE_HOME*\PRO80\C\INCLUDE

directory to the directory that contains the problematic include file and run MOD_
INCL.BAT.

Using the Precompiler Options

Chapter 9 of the *Pro*C/C++ Precompiler Programmer's Guide* describes the Pro*C/C++ options. This section highlights issues that are related to Pro*C/C++ for Windows NT and Windows 95/98.

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 deprecated. Use CHAR_MAP=VARCHAR2 with DBMS=V7 instead.

INCLUDE

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 "includes=" 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 at a different location or if Borland C++ is used, 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.

VER

The VER option is a Windows-specific option.

It takes the first and second most significant digits of the version number of Pro*C/C++ you want to use. This option is an alternative to changing the PROC_DFLT value in the registry.

For a previously installed Pro*C/C++ 2.2 to precompile SAMPLE1.PC, for example, type the following:

```
PROC.EXE VER=22 SAMPLE1.PC
```

Note: This option can only be used to execute previously installed Pro*C/C++ release 2.2.2.0.1 or later. Simply running `proc.exe` will execute the most recently installed version of Pro*C/C++.

PROC_DFLT

When Pro*C/C++ is installed, PROC_DFLT is set to PROC80 by default in the registry. If multiple versions of Pro*C/C++ are installed, simply resetting the value to a valid Pro*C/C++ version number will allow a different version to be the default when `proc.exe` is executed at the MS-DOS command prompt. For example, setting PROC_DFLT to PROC22 will make Pro*C/C++ 2.2 the default. This feature applies only to command line Pro*C/C++.

3 Building Pro*C/C++ Applications

This chapter 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.

Specific topics discussed are:

- Building and Running Sample Programs
- Building Multi-Threaded Applications

Building and Running Sample Programs

When you install Pro*C/C++, Oracle Installer copies a set of sample programs to the *ORACLE_HOME*\PRO80\C\SAMPLES 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.

Note: The *Pro*C/C++ Precompiler Programmer's Guide* indicates that the sample programs are located in a directory named DEMO. In Pro*C/C++ for Windows NT and Windows 95/98, the sample programs are located in the directory named SAMPLES.

Directory	Sample Program	Sample Pro*C/C++ Project	Sampler Compiler Project	Description
COLDEMO1	COLDEMO1.PC COLDEMO1.SQL	COLDEMO1.PRE	COLDEMO1.DSP COLDEMO1.IDE	Pro*C/C++ application that uses the Object Type Translator (OTT). **Note:** Run COLDEMO1.SQL and OTT before building COLDEMO1.
CPPDEMO1	CPPDEMO1.PC	CPPDEMO1.PRE	CPPDEMO1.DSP CPPDEMO1.IDE	Pro*C/C++ application in C++.
CPPDEMO2	CPPDEMO2.PC	CPPDEMO2.PRE	CPPDEMO2.DSP CPPDEMO2.IDE	Pro*C/C++ application in C++.
CPPDEMO3	CPPDEMO3.PC	CPPDEMO3.PRE	CPPDEMO3.DSP CPPDEMO3.IDE	Pro*C/C++ application in C++.
CVDEMO	CV_DEMO.PC CV_DEMO.SQL	CV_DEMO.PRE	CV_DEMO.DSP CV_DEMO.IDE	Pro*C/C++ application using cursor variable with PL/SQL stored procedure. **Note:** Run CV_DEMO.SQL before building CV_DEMO.
LOBDEMO1	LOBDEMO1.PC LOBDEMO1.SQL	LOBDEMO1.PRE	LOBDEMO1.DSP LOBDEMO1.IDE	Pro*C/C++ application that uses large objects (LOBs). **Note:** Run LOBDEMO1.SQL before building LOBDEMO1.

Directory	Sample Program	Sample Pro*C/C++ Project	Sampler Compiler Project	Description
MLTTHRD1	MLTTHRD1.PC MLTTHRD1.SQL	MLTTHRD1.PRE	MLTTHRD1.DSP MLTTHRD1.IDE	Multi-threaded Pro*C/C++ application. **Note:** Run MLTTHRD1.SQL before building MLTTHRD1.
NAVDEMO1	NAVDEMO1.PC NAVDEMO1.SQL	NAVDEMO1.PRE	NAVDEMO1.DSP NAVDEMO1.IDE	Pro*C/C++ application that demonstrates navigational access to objects in the object cache. **Note:** Run NAVDEMO1.SQL and OTT before building NAVDEMO1.
OBJDEMO1	OBJDEMO1.PC OBJDEMO1.SQL	OBJDEMO1.PRE	OBJDEMO1.DSP OBJDEMO1.IDE	Pro*C/C++ application that uses OTT. **Note:** Run OBJDEMO1.SQL and OTT before building OBJDEMO1.
ORACA	ORACA.PC ORACATST.SQL	ORACA.PRE	ORACA.DSP ORACA.IDE	Pro*C/C++ application using the ORACA to determine various performance parameters at runtime. **Note:** Run ORACATST.SQL before building ORACA.
PLSSAM	PLSSAM.PC	PLSSAM.PRE	PLSSAM.DSP PLSSAM.IDE	Pro*C/C++ application with embedded PL/SQL blocks.
SAMPLE	SAMPLE.PC	SAMPLE.PRE	SAMPLE.DSP SAMPLE.IDE	Pro*C/C++ application using EXEC SQL statements.
SAMPLE1	SAMPLE1.PC	SAMPLE1.PRE	SAMPLE1.DSP SAMPLE1.IDE	Pro*C/C++ application using EXEC SQL statements.
SAMPLE2	SAMPLE2.PC	SAMPLE2.PRE	SAMPLE2.DSP SAMPLE2.IDE	Pro*C/C++ application using EXEC SQL statements.
SAMPLE3	SAMPLE3.PC	SAMPLE3.PRE	SAMPLE3.DSP SAMPLE3.IDE	Pro*C/C++ application using EXEC SQL statements.
SAMPLE4	SAMPLE4.PC	SAMPLE4.PRE	SAMPLE4.DSP SAMPLE4.IDE	Pro*C/C++ application that demonstrates type equivalencing.

Directory	Sample Program	Sample Pro*C/C++ Project	Sampler Compiler Project	Description
SAMPLE5	SAMPLE5.PC EXAMPBLD.SQL EXAMPLOD.SQL	SAMPLE5.PRE	SAMPLE5.DSP SAMPLE5.IDE	Pro*C/C++ application with embedded PL/SQL blocks. **Note:** Run EXAMPBLD.SQL, then run EXAMPLOD.SQL, before building SAMPLE5.
SAMPLE6	SAMPLE6.PC	SAMPLE6.PRE	SAMPLE6.DSP SAMPLE6.IDE	Pro*C/C++ application using EXEC SQL statements.
SAMPLE7	SAMPLE7.PC	SAMPLE7.PRE	SAMPLE7.DSP SAMPLE7.IDE	Pro*C/C++ application using EXEC SQL statements.
SAMPLE8	SAMPLE8.PC	SAMPLE8.PRE	SAMPLE8.DSP SAMPLE8.IDE	Pro*C/C++ application using EXEC SQL statements.
SAMPLE9	SAMPLE9.PC CALLDEMO.SQL	SAMPLE9.PRE	SAMPLE9.DSP SAMPLE9.IDE	Pro*C/C++ application calling a PL/SQL stored procedure. **Note:** Run CALLDEMO.SQL before building SAMPLE9.
SAMPLE10	SAMPLE10.PC	SAMPLE10.PRE	SAMPLE10.DSP SAMPLE10.IDE	Pro*C/C++ application showing an implementation of Dynamic SQL Method 4.
SAMPLE11	SAMPLE11.PC SAMPLE11.SQL	SAMPLE11.PRE	SAMPLE11.DSP SAMPLE11.IDE	Pro*C/C++ application that uses a PL/SQL stored procedure. **Note:** Run SAMPLE11.SQL before building SAMPLE11.
SAMPLE12	SAMPLE12.PC	SAMPLE12.PRE	SAMPLE12.DSP SAMPLE12.IDE	Pro*C/C++ application that demonstrates Dynamic SQL Method 4.
SQLVCP	SQLVCP.PC	SQLVCP.PRE	SQLVCP.DSP SQLVCP.IDE	Pro*C/C++ application that demonstrates the use of sqlvcp() and sqlgls() calls.
WINSAM	WINSAM.PC	WINSAM.PRE	RESOURCE.H WINSAM.DSP WINSAM.H WINSAM.ICO WINSAM.IDE WINSAM.RC	GUI Pro*C/C++ application.

.DSP and .IDE Files

.DSP and .IDE files help you build application programs.

- .DSP files guide and control the steps necessary to precompile, compile, and link the Pro*C/C++ sample programs.
- .IDE files guide and control the steps necessary to compile and link the Pro*C/C++ sample programs.

Sample .DSP files in this release were created with Microsoft Visual C++ version 5.0. Sample .IDE files in this release were created with Borland C++ version 5.0.

Sample Pro*C/C++ Programs

When built, the sample programs that Oracle provides in Pro*C/C++ produce .EXE executables. For some sample programs, 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.

Also, some samples require the Object Type Translator (OTT) to be executed before precompilation. Use OTT to create header files with C-struct representations of Abstract Data Types that have been created and stored in an Oracle8 database.

Additional Information: For more information on OTT, see Chapter 16 of the *Pro*C/C++ Precompiler Programmer's Guide*.

The sample programs are described below.

COLDEMO1.PC

Fetches census information for California counties. This program demonstrates a number of ways to navigate through collection-typed database columns.

CPPDEMO1.PC

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

CPPDEMO2.PC

Retrieves the names of all employees in a given department from the EMP table (Dynamic SQL Method 3).

CPPDEMO3.PC

An example of C++ inheritance, this program finds all salespeople and prints their names and total earnings (including commissions).

CV_DEMO.PC

Fetches data 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.

LOBDEMO1.PC

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 via the DBMS_LOB package.

MLTTHRD1.PC

Shows how to use threading in conjunction with precompilers. The program creates as many sessions as there are threads.

NAVDEMO1.PC

Demonstrates navigational access to objects in the object cache.

OBJDEMO1.PC

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

ORACA.PC

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

PLSSAM.PC

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.PC

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.PC

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 you enter 0 as the employee number.

SAMPLE2.PC

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.PC

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.PC

Demonstrates the use of type equivalencing using the LONG RAW external datatype.

SAMPLE5.PC

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.

SAMPLE6.PC

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

SAMPLE7.PC

Inserts two rows into the EMP table and deletes them (Dynamic SQL Method 2).

SAMPLE8.PC

Retrieves the names of all employees in a given department from the EMP table (Dynamic SQL Method 3).

SAMPLE9.PC

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.PC

Connects to an Oracle database using your user name 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.PC

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.PC

This is a very simple program that demonstrates how to do array fetches using dynamic SQL Method 4.

SQLVCP.PC

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 *sqlgls()* function to retrieve the text of the last SQL statement that was executed.

WINSAM.PC

This GUI version of the SAMPLE.PC program 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 box.

Creating a Sample Program

To precompile, compile, and link a sample program:

1. Open a sample project file. For example, open WINSAM.DSP for Microsoft Visual C++ or WINSAM.IDE for Borland C++.
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.

   Note: All of the sample projects were created with C as the default drive.

   Attention: If you are using Microsoft Visual C++, note that SQL*Plus, OTT, and Pro*C/C++ have been fully integrated into the Microsoft Visual C++ sample project files. See Appendix A, "Integrating Pro*C/C++ into Microsoft Visual C++". You do not have to run SQL*Plus, OTT, and Pro*C/C++ separately before compilation. Simply open the project file and choose Rebuild All from the Build menu to create the sample program using Microsoft Visual C++. The remaining steps in this procedure can then be omitted.

   Attention: If you are using Borland C++ version 5.0, you can precompile all of the samples using Pro*C/C++ as a tool from the Tool menu. See Appendix B, "Integrating Pro*C/C++ into Borland C++". Continue with the remaining steps in this procedure.

   If you use Borland C++, some samples require SQL scripts and OTT to be executed individually before precompilation.

3. For samples that require SQL scripts to be executed, enter the following at the MS-DOS command prompt:

   ```
   sqlplus @script.sql
   ```

where *script* is the name of a SQL script for a sample. SQL scripts for samples can be found in the corresponding sample directory. See the table earlier in this chapter for the name of the SQL script that needs to be executed for a particular sample, if any.

4. For the samples that require OTT to be executed, enter the following at the command line to produce .H and intype files for Pro*C/C++:

   ```
   ott intype=demo.typ outtype=demo_o.typ hfile=demo.h code=c
   userid=scott/tiger
   ```

   where *demo* is `objdemo1`, for example. When precompiling samples that require OTT, be sure to set `intype=`*demo*`_o.typ`, `parse=full`, and include directory to the location of *demo*`.h`.

   Additional Information: See Chapter 16 of the *Pro*C/C++ Precompiler Programmer's Guide* for more information about OTT.

5. Choose Pro*C/C++ from the Tool menu.

   The Pro*C/C++ precompile environment appears with the project file (for example, WINSAM.PRE) already loaded.

6. Click Precompile on the Pro*C/C++ toolbar to precompile the sample file.
7. Once the precompile process is successfully completed, go back to your development environment and build the sample application.

Running a Sample Program

To run a sample program (in this case, WINSAM) after building it:

1. From your development environment, run WINSAM.EXE.
2. Choose Connect from the Oracle menu.
3. Enter SCOTT in the User ID box and TIGER in the Password box (or the connect string if connecting remotely).
4. Click OK.

   A dialog box appears notifying you whether the connection is successful.

5. To enter employee data, choose Employees from the Oracle menu.

An *Employee Record* dialog box appears. You can use SQL buttons such as SELECT, FETCH, or INSERT to manipulate the sample tables.

6. Choose Exit from the *Employee Record* dialog box. Then, choose Disconnect from the Oracle menu. A dialog box notifies you whether the disconnect is successful.
7. To quit the program, choose Exit.

Note: All sample programs are written to be run on the local database. To connect to a remote database, use the LOCAL variable in the registry.

Building the Demonstration Tables

To run the Pro*C/C++ sample programs, you must have a database account with the user name SCOTT and the password TIGER and a database with the demonstration tables EMP and DEPT. This account is included in the seed database (named ORACLE) that is included with your Oracle server. If no such account exists on your database, install one before running the sample programs.

Additional Information: See your Oracle database or server documentation. If your database does not contain these tables, use the DEMOBLD.SQL script to create them. To run DEMOBLD, first use SQL*Plus to connect to the Oracle database.

Dynamic Link Libraries (DLLs)

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 workstation that is running your Pro*C/C++ application.

Applications linked with DLLs provide the following benefits over static libraries:

- Multiple applications can share the same DLL module, conserving disk space and memory.
- You can update DLL modules by overwriting the DLL files. When you do this, you do not have to relink applications that access the DLL module.

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.

Building Multi-Threaded Applications

Build multi-threaded applications if you are planning to perform concurrent database operations.

Windows NT 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. Multi-threaded applications use the resources of a process to coordinate the activities of individual threads.

Additional Information: See the *Pro*C/C++ Precompiler Programmer's Guide* for additional information on how to write multi-threaded applications with Pro*C/C++.

When building a multi-threaded 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 multi-threaded 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.

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.

Specific topics discussed are:

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

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

Microsoft Visual C++ maintains the dependency and precompile files, if needed. 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.

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

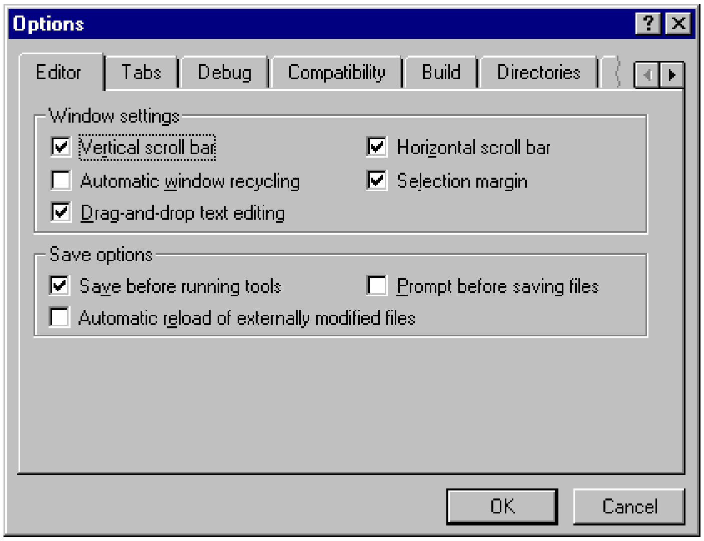
Specifying the Location of the Pro*C/C++ Executable

For Microsoft Visual C++ to run Pro*C/C++, it needs to know the location of the Pro*C/C++ executable. If Microsoft Visual C++ was installed before any Oracle 8.0 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.


Options

Window settings
- [✓] Vertical scroll bar
- [✓] Horizontal scroll bar
- [ ] Automatic window recycling
- [✓] Selection margin
- [✓] Drag-and-drop text editing

Save options
- [✓] Save before running tools
- [ ] Prompt before saving files
- [ ] Automatic reload of externally modified files

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_HOME*\BIN directory. For example:

   `C:\ORANT\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_HOME*\PRO80\C\INCLUDE directory. For example:

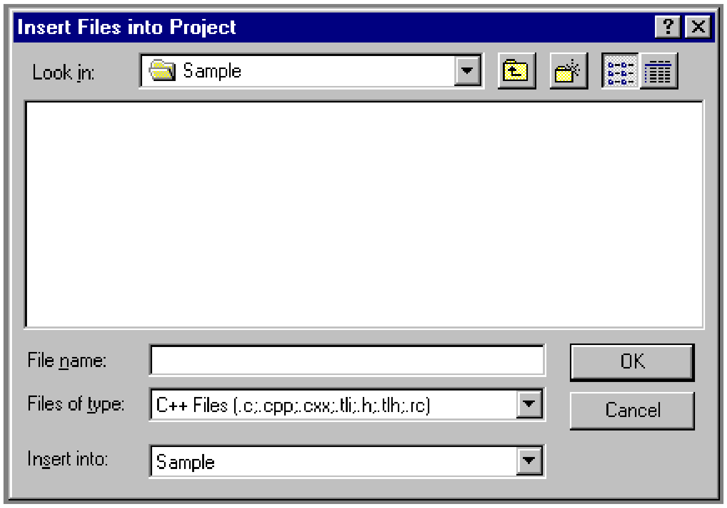
   `C:\ORANT\PRO80\C\INCLUDE`
6. Click OK.

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.


Insert Files into Project
Look in:
Sample
File name:
Files of type:
C++ Files (.c;.cpp;.cxx;.tli;.h;.tlh;.rc)
Insert into:
Sample
OK
Cancel

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 SQLLIB80.LIB.

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

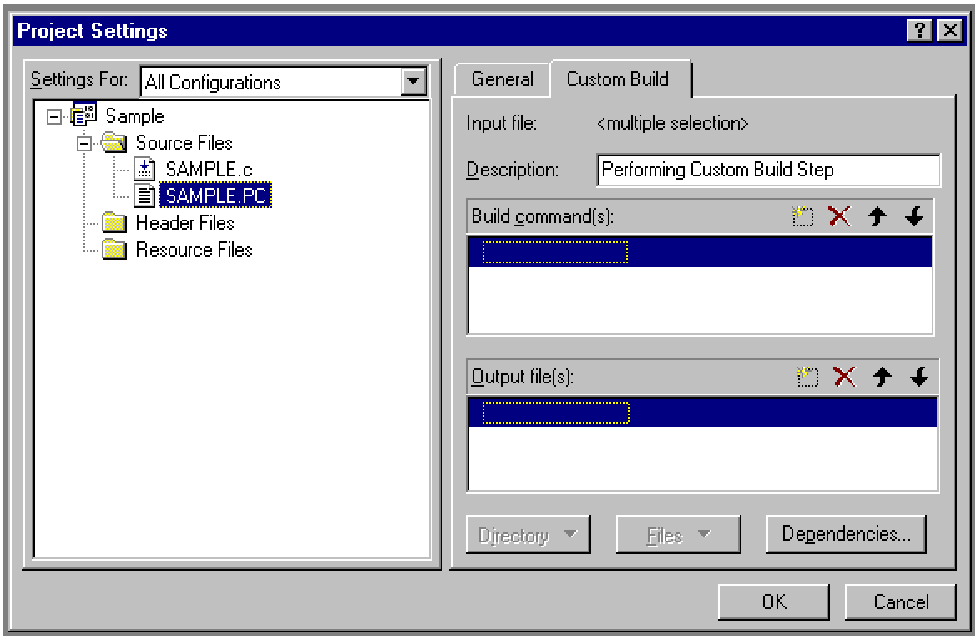
1. Choose Add To Project from the Project menu, and then choose Files.

   The *Insert Files into Project* dialog box appears.
2. Select All Files from the Files of type list box.
3. Select SQLLIB80.LIB from the *ORACLE_HOME*\PRO80\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.



Project Settings

Settings For: All Configurations

Sample
Source Files
SAMPLE.c
SAMPLE.PC
Header Files
Resource Files

General | Custom Build

Input file: <multiple selection>

Description: Performing Custom Build Step

Build command(s):

Output file(s):

2. In the Build command(s) box, enter the following on one line:

```
$(ProjDir)\..\..\..\..\bin\proc $(ProjDir)\$(InputName).pc
include=$(ProjDir)\..\..include include="$(MSDEVDIR)\..\vc\include"
```

   $(ProjDir) and $MSDEVDIR are macros for custom build commands in Microsoft Visual C++. See the Microsoft Visual C++ documentation for more information.

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

   - `$(ProjDir)\$(InputName).c` (if you are generating a .C file)
   - `$(ProjDir)\$(InputName).cpp` (if you are generating a .CPP file)

   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.

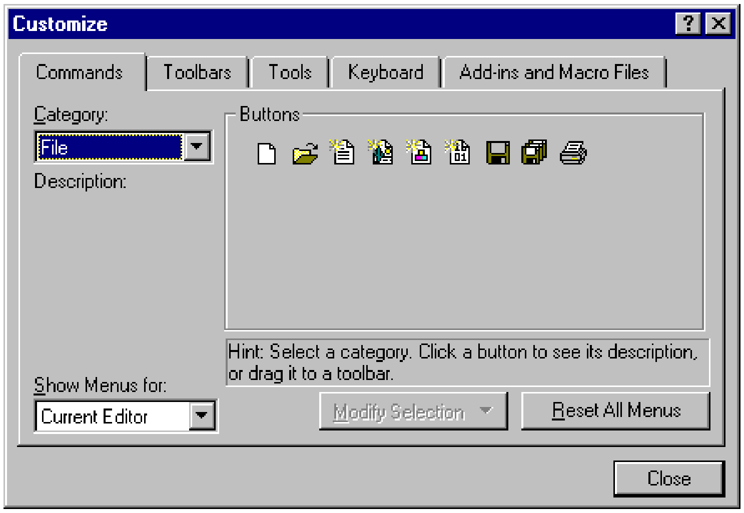
4. Click OK.

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.



Hint: Select a category. Click a button to see its description, or drag it to a toolbar.

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++ 8.0

5. In the Command box, type the path and file name 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:\ORANT\BIN\PROCUI80.EXE
```

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

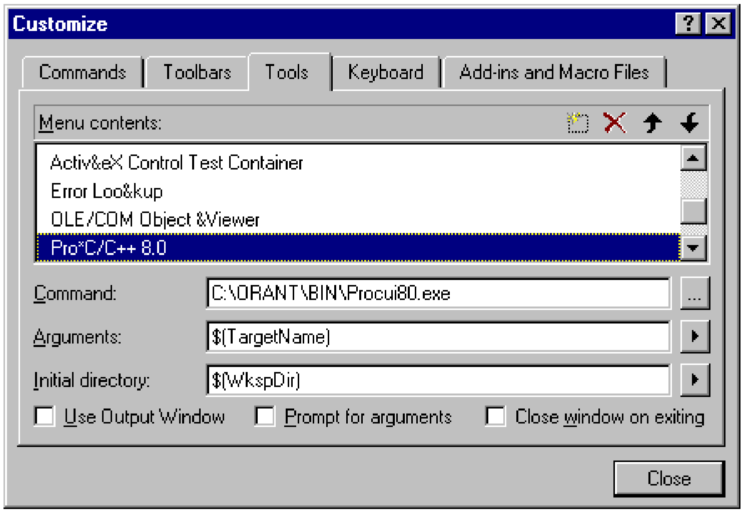
   ```
   $(TargetName)
   ```

   When you choose Pro*C/C++ 8.0 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).



Command: C:\ORANT\BIN\Procui80.exe
Arguments: $(TargetName)
Initial directory: $(WkspDir)

8. Click Close.

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

B

Integrating Pro*C/C++ into Borland C++

This appendix describes how to integrate Pro*C/C++ into the Borland C++ version 5.0 integrated development environment. The specific topic discussed is:

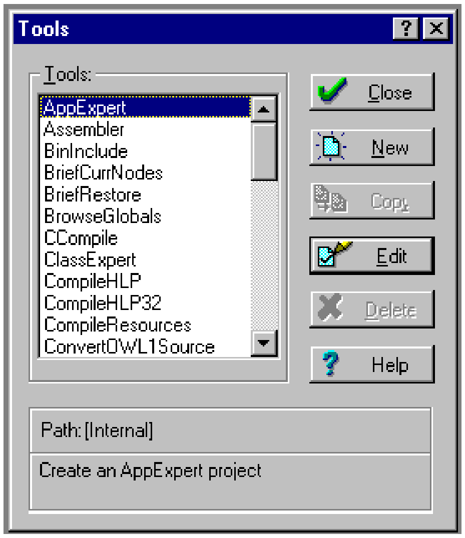
- Adding Pro*C/C++ to the Tool Menu

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

You can include Pro*C/C++ as a choice in the Tool menu of Borland C++.

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

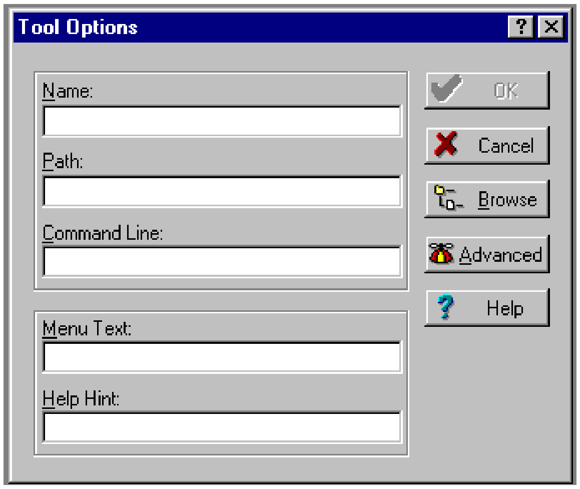
1. From within Borland C++, open a project file.
2. Choose Tools from the Options menu. The *Tools* dialog box appears.



Path: [Internal]

Create an AppExpert project

3. Click New. The *Tool Options* dialog box appears.



4. In the Name box, enter the following text:

```
Pro*C/C++ 8.0
```

5. In the Path box, enter the following text:

```
PROCUI80.EXE
```

If the PATH environment variable does not include your *ORACLE_HOME*\BIN directory, enter the path before the file name. For example:

```
C:\ORANT\BIN\PROCUI80.EXE
```

6. In the Command Line box, enter the following text:

```
$PRJNAME
```

When you choose Pro*C/C++ 8.0 from the Tool menu, Borland C++ uses the $PRJNAME 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 but with a .PRE extension in the project directory.

7. In the Menu Text box, enter the following text:

```
&Pro*C/C++ 8.0
```

The ampersand causes Borland C++ to assign a shortcut key to the letter P.

8. In the Help Hint box, enter the following text:

```
Precompile
```

This text appears when you highlight Pro*C/C++ 8.0 on the Tool menu.

9. Click OK in the *Tool Options* dialog box.
10. Click Close in the *Tools* dialog box.

    Borland C++ adds Pro*C/C++ to the Tool menu.
11. Save the project file.

Index