Welcome to C++Builder Client/Server Suite

We welcome you to the C++Builder Client Server Suite!

This ReadMe contains the following topics:

- Contacting Borland
- C++Builder product features
- Important information
- Manual corrections

To view these topics, click the Browse buttons at the top of the Help window.
>> moves you forward through the Help file, and **<<** moves you backward.

The C++Builder Development Team
Borland International

Contacting Borland

Borland offers a range of services to provide you with the most comprehensive product information, updates, and product service. For instance, you can get more information on C++Builder by contacting any of these resources:

World Wide Web:	http://www.borland.com/
CompuServe:	GO BCPP, section 3, "From Borland"
FTP:	ftp.borland.com
BBS:	(408) 431-5096. 8-N-1 (I bits, No parity, 1 stop bit)
Install (Up & Running)	
Listserv:	listserv@borland.com Send an email message containing this text:

```
SUBSCRIBE<space>BCPP<space>FIRSTNAME<space>LASTNAME
```

Borland Online

Although we offer a complete range of support services, don't forget to first point your World Wide Web browser to Borland Online:

http://www.borland.com/

The Borland C++Builder Product Team will deliver regular posts of useful product information including White Papers, competitive analyses, answers to common questions, sample applications, and continuing updates on our product development progress.

You can also visit these Borland URL sites for more information:

Tech Info page:	http://www.borland.com/TechInfo/
C++ page:	http://www.borland.com/TechInfo/cpp

- If you still need assistance, help is available from the following sources:

Developer Support

Borland offers a set of Developer Support plans, ranging from general product usage and installation to the specifics of C++Builder language syntax, programming, and debugging.
For information about these support programs, call 1-800-523-7070.

C++ Programmer's Advisor Lines

For immediate assistance with everything from configuring C++Builder to C++ programming or debugging, call our C++ Programmer's Advisor Lines ($2.95/minute, first minute free):

- Windows / Win32: 1-800-782-5558 (MC/VISA)

For assistance outside North America, see the section below, "Help outside North America," or contact your local Borland representative.

CompuServe

For online access to the Borland CompuServe forums, with their libraries of technical information and answers to common questions, type:

GO BCPP	For questions related to C/C++ programming languages and Borland's programming tools
GO BDEVTOOLS	For questions about the Borland Database Engine
GO BCPPLIB	For questions about ObjectWindows Library, STL, and other Borland class libraries, and the Windows API.

If you are not a member of CompuServe, see the enclosed special offer, and write for full details on how to receive a free IntroPak containing a $15 credit toward your first month's online charges.

TECHFAX

Call Borland's TECHFAX service at 1-800-822-4269 for a FAX catalog of technical document entries. For assistance outside North America, contact your local Borland representative.

Borland Data Library
You can contact the Borland DLBBS by dialing (408) 431-5096 (up to 9600 baud, 8-N-1) for a host of free technical documents and example programs.

FTP site
Technical information on Borland C++Builder is available on the Internet via anonymous ftp at our ftp site ftp.borland.com.

Borland Newsletters
Subscribe to Borland's free electronic newsletter and get regular updates on up-to-date technical tips, patch notifications, bug fixes, and product releases. Send your full name and address via electronic mail to tech-info@borland.com.

Help outside North America
For help outside North America, contact any of the following telephone numbers:

Australia	1 800 641 144
Austria	+49 (0) 8995914705
Belgium (NL)	+32 (0) 27298022
Belgium (FR)	+32 (0) 27298035
Czech Republic	+42 (2) 6272135
Denmark	+45 (0) 45762313
Finland	+358 (09) 04209792
France	+33 (1) 41377019
Germany	+49 (0) 8995914705
Iceland	+47 (0) 22250017
Ireland	+44 (0) 1256373479
Italy	+39 (2) 57303203
Netherlands	+31 (0) 30 833730
Norway	+47 (0) 22250017
Portugal	+34 (1) 6618091
South Africa	+27 11 7894316
Spain	+34 (1) 6618091
Sweden	+46 (0) 86297520
Switzerland	+49 (0) 8995914705
UK	+44 (0) 1256373479

C++Builder product features

This section contains information on the following subjects:

- C++Builder interface and components
- Examples
- Help files
- Differences between C++Builder and Delphi
- C++ compiler modifications
- Object Pascal compiler modifications

C++Builder interface and components

C++Builder provides a rapid application development (RAD) environment and tools to help you create 32-bit Windows applications. It combines the visual development environment of Delphi with cached, pre-compiled headers and incremental/smart linking to provide fast-turnaround cycles while developing projects. C++Builder also provides a visual development environment with database connectivity.

C++Builder's design incorporates most Delphi interface and development components, but generates C++ code for projects, forms, units, and database modules. C++Builder contains two compilers, a C++ compiler and an Object Pascal compiler. Along with the code generated by C++Builder, the C++ compiler accepts all the code you have created with Borland C++ 5.0x. Similarly, the Object Pascal compiler compiles all Object Pascal code generated by Delphi 2.0.

In addition, C++Builder uses the same Visual Component Library (VCL) components as Delphi, which you will find on the C++Builder Component palette.

Examples

C++Builder comes with a multitude of example programs to help you get started. Use the File|Open Project command to open any of the .MAK project files located under the Examples directory, then press F9 to build and run the example.

- Note that for AutoCon.mak to build and run correctly, you must first build and run the "AutoSrv" example.

Help files

C++Builder has a rich set of Help files to supplement the printed manuals. In particular, the BCBVCL.HLP files offers a complete online reference to the VCL. The online file BCBPG.HLP gives you the material contained in the Programmer's Guide, and BCB.HLP gives you a complete reference to the C++Builder environment. For help on creating your own components, see the material contained in BCBCWG.HLP, and for database programming, see BCBDADG.HLP.

Differences between C++Builder and Delphi

Note: C++Builder can use Delphi forms, source code, and data modules through the Project | Add to Project menu option. To create new components or subclass from a Delphi component, select New Component from the Object Repository.

In C++Builder, project options are kept in project "makefiles," which end with a .MAK file extension. All compiler/linker options for a C++Builder project are stored in this fully functioning makefile that can be run outside C++Builder using the supplied command line tools. The project makefiles can be edited to change or add advanced compiler/linker options not supported by the UI; however, if you want to continue to use the project file with C++Builder, you must conform to the C++Builder makefile syntax.

The C++Builder Component Library is based on object files (.OBJs) rather than the Delphi compiled units (.DCU files) that Delphi 2.0 uses. With C++Builder, you can mix .PAS and .CPP units in the same .DLL.

C++ compiler modifications

Native properties

Native properties have been added. An example:

```
__property String FileName =
  {read=GetFileName, write=SetFileName};
```

```
  // ...
  FileName += ".cpp";
//  or
  FileName += "MySource" + ".cpp";
```

Native closures

Native closures are supported. An example:

```
  void (__closure *OnClickButton) (TObject* Sender);
  // ...
  OnClickButton( foo );  // call handler
  OnClickButton = bar->MyHandleButton;  // set new handler
  OnClickButton( foo );  // call different handler
```

See the minicomp example for usage of properties and closures. In addition, refer to the BCBCWG.HLP Help file for more information on properties and events (closures).

Keywords

__declspec (delphiclass | delphireturn)

__automated

__published

__closure

__property

__classid (class)

__dispid (int)

Other C++ additions

The following new classes have been added to the C++Builder libraries: *AnsiString, Variant, ShortString, Currency, TDateTime,* and *Set*.

The C++Builder runtime library provides functions which let you develop locale-sensitive applications. To support locale-sensitive applications, the RTL now contains functions that have *wchar_t* and multi-byte arguments. See the BCBRTL.HLP online Help files for the RTL details.

The following new open array helper macros have been added to support interacting with VCL objects:

- OPENARRAY
- ARRAYOFCONST
- EXISTINGARRAY
- SLICE

Also, support has been added for:

- Calling and defining dynamic and message functions
- OLE automation controllers with variant dispatching

The following new switches have been added to the C++Builder C++ compiler (BCC32.EXE):

- **-Vx** switch for truly empty (0 length) structs.
- -**He** switch (extern types in .objs)
- **-Hs** switch (smart cached headers)

The following new **#pragma**s have been added to the C++ compiler:

- #pragma link "obj" (object file dependencies)
- #pragma resource "res" (resource file dependencies)
- #pragma anon_struct on (for support of VCL variant records)

Support for dynamic_cast<> of VCL objects has been added.

Object Pascal compiler modifications

The following new switches have been added to the Object Pascal compiler (DCC32.EXE):

- **-jp** switch: creates Borland C++ compatible .obj files.
- **-jph** switch: creates C++Builder compatible header (.hpp) files from Object Pascal unit files (.dcl).
- **-jphn** switch: uses the Object Pascal unit name as the enclosing C++ namespace for both .objs and .hpps that are generated.
- **-n** switch: specify .dcu output directory
- **-nh** switch: specify .hpp output directory
- **-no** switch: specify .obj output directory

C++Builder does not support old-style Object, Real, and Comp types in the Interface section of an Object Pascal unit. Additionally, although Comp is semi-supported as a non-numeric type, it should not be used; use Currency instead.

Important information

Please read the following items before using C++Builder:

- If you are running Windows 95, you cannot debug your program from an in-memory executable image (Windows 95 does not support the option Options | Project | Linker | In-Memory .EXEs).
- If you are going to install any sample controls from the C++Builder Examples\Controls directories, make sure the Library options are correctly set up before you install the components. You can set the library options from the Environment Options dialog box (Options|Environment).
- You cannot catch, then throw, a VCL exception with a catch block that uses a named argument. Instead, use an unnamed catch block for the exception. For example:

```
...
   try
   {
       throw Exception("foo");
   }
   catch(...)            // Works
// catch(Exception & e) // Rethrow throws garbage
// catch(Exception &)   // Rethrow throws garbage
   {
       throw;
   }
...
```

Manual corrections

Because software manuals need to go to the printer several weeks before the end of the product development cycle, software manuals usually contain a few errors. The following is an errata for the C++Builder printed manuals:

Currently, there are no know errors in the C++Builder manuals.