Welcome to the Borland C++Builder Trial edition

Welcome to the Borland C++Builder Trial edition!

You may use this free trial version for sixty days or until 12/31/97 (whichever comes first) to try out Borland C++Builder. After the trial period has ended, the Borland C++Builder Trial edition will be disabled. To continue using C++Builder after the trial period, you can purchase one of the following full-featured versions:

- C++Builder Client/Server Suite
- C++Builder Professional edition
- C++Builder Standard edition

For more information, see Contacting Borland.

Register your Borland C++Builder online!

If you have Internet access and an installed browser,

click here {button Register Borland C++Builder,EF(`http://www.borland.com/bcppbuilder/bcbwebreg.html',`',1,`')} to register your copy so that you will be notified of the latest Borland C++Builder product information and product offerings.

This ReadMe contains the following topics:

- Contacting Borland
- C++Builder product features
- Important information
- Errata for printed documentation
- Additional information about C++Builder

To view the topics sequentially, 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 (1 bits, No parity, 1 stop bit)
Up & Running	(408) 461-9133 Support to help you get C++Builder installed on your system. No charge to registered users.
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:

C++Builder page:	http://www.borland.com/bcppbuilder
Tech Info page:	http://www.borland.com/techsupport/borlandcpp

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 the VCL, Borland C++Builder, and C++ libraries

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 folder, then press F9 to build and run the example.

- For specific information on each example, please refer to the ReadMe.TXT file in each example folder. For example, for AutoCon.mak to build and run correctly, you must first build and run the "AutoSrv" example. In addition, you must also install a sample component (or components) to compile some of the example programs.

Help files

C++Builder has a rich set of Help files to supplement the printed manuals. In particular, the BCBVCL.HLP file 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

- C++Builder can use Delphi forms, source code, and data modules through the Project|Add to Project menu option. To create new components or to subclass from existing VCL components, use the Component Wizard (which you can access from the Component|New command on the main menu or New Component command in the Object Repository).

In C++Builder, project options are kept in project "makefiles," which end with a .MAK file extension.

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 = 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 | hidesbase | pascalimplementation)

__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:

- **-CP** enables user-defined code paging (to support MBCS)
- **-He** enables pre-compiled headers with external-type .OBJ files
- **-Hs** enables smart cached pre-compiled header files
- **-V?** places all functions into VIRDEFs for smart linking
- **-Ve** enables support for empty base classes
- **-Vl** backward compatibility switch; uses old-style structure layout
- **-Vx** for truly empty (0 length) structs
- **-xdg** backward compatibility switch; uses global destructor count
- **-WU** enables support for UNICODE

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 pointers to VCL objects has been added.

Object Pascal compiler modifications

The following new switches have been added to the Object Pascal compiler:

- **`-jp`** switch: creates Borland C++ compatible .obj files.
- **`-jph`** switch: creates C++Builder compatible header (.hpp) files from Object Pascal source files (.pas).
- **`-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 folder
- **`-nh`** switch: specify .hpp output folder
- **`-no`** switch: specify .obj output folder

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, you should avoid using it; use Currency instead.

Important information

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

- If you are running Windows 95, you cannot run your program from an in-memory executable image (Windows 95 does not support the option Options|Project|Linker|In-Memory .EXEs).
- Before running a sample application, please review the associated README.TXT for any setup requirements. The README.TXT file for the sample application, if one exists, can be found in the same folder as the sample application's source files.
- Before you install a sample component, be sure to first see README.TXT located by default in the C++Builder Examples\Controls folder.
- If you experience any problems with the C++Builder Help systems under Windows 95, you should download the latest version of WINHLP32.EXE from Microsoft's Web site (http://www.microsoft.com).
- By default, all new projects are saved in the CBuilder\Projects folder. If you want to change the default location, specify a different **Start In** path for the BCB.EXE shortcut (or create a new shortcut to BCB.EXE) and choose Start|Settings|Taskbar|Advanced, then right-click the CBuilder item, and choose Properties.
- To create an application that uses the dynamic version of the RTL, you must:
  1. Check the Use Dynamic RTL on the Linker page of the Project Options dialog box.
  2. Define **_RTLDLL** in the Conditional Defines input box on the Directories/Conditionals page of the Project Options dialog box (use a semi-colon to separate multiple defines).

  Be sure to reverse these two actions to change back to a statically-linked application.
- Do not move your CBuilder\Include and CBuilder\Lib folders from their installation locations. If you move these folders, your projects may not compile correctly.
- If you are using TASM32.EXE directly, or if the source code you are compiling contains inline assembler statements, and you encounter an error such as :

  ```
  [Linker Fatal Error] Fatal: Bad object file 'project1.cpp' near file offset 318d
  ```

  you need to update your version of TASM32.EXE. To do so, run Install.BAT located in the \TASM_TD folder on the C++Builder installation CD.

  Install.BAT takes two parameters:

  ```
  Install <C++Builder_Drive> <C++Builder_Folder>
  ```

  For example, if you installed C++Builder in C:\CBUILDER, you would enter (be sure to include the backslash in the second parameter):

  ```
  Install  c:  \cbuilder
  ```
- If you are running C++Builder under Windows 95 while using the incremental linker and you run out of disk space, you may not be able to continue to link during the current Windows session.

  Because of a problem related to Windows 95's memory mapped file implementation, even after if you free enough disk space, the incremental linker will continue to report (based on information it receives back from the operating system) that there is not enough disk space to build the linker's state files and the target executable.

  If this situation occurs, you have the following options:

  - Disable incremental linking for the current Windows session (choose Options|Project|Linker| and clear the Incremental linking checkbox)
  - Save the project to a different drive and build it there.
  - Save your project and restart Windows and C++Builder.
- Async Professional for Delphi 2.01 by TurboPower is incompatible with C++Builder Version 1.0. Installing this component will result in the following error:

  ```
  Unsupported language feature: 'Real'
  ```

Errata for printed documentation

Because software manuals need to go to the printer several weeks before the end of the product development cycle, they can sometimes be out of date when compared with the shipping software. The following is an errata for the C++Builder printed manuals:

C++Builder Visual Component Library Reference

- Some of the documentation for the database objects, including TStoredProc and TQuery, refer to a TBDEDataSet object. This object does not exist in the C++Builder version of the Visual Component Library.
- The documentation for the TFloatField, TCurrencyField, and TBCDField objects incorrectly refers to the “Currency” property. This property is actually the “currency” property (with a lower-cased ‘c’). By using a lower-cased ‘c’, you can distinguish the “currency” property from the “Currency” class defined in the system.hpp file.
- The documentation for the TWinControl object is missing descriptions of the ImeName and ImeMode properties.
- The documentation for the TFont object is missing a description of the Charset property.
- The TReport object does not exist in the C++Builder version of the Visual Component Library. For information on the Quick Reports reporting tools, see the QUICKREP.HLP file and the QREPORT.DOC document file.

Additional information about Borland C++Builder

The following topics contain additional information about Borland C++Builder not described elsewhere in the Version 1.0 online or printed documentation.

Default project files

TFont::Charset

TWinControl::ImeMode

TWinControl::ImeName

Enabling UNICODE in your C++Builder application

Pascal interface procedure support in C++Builder

Distributing applications not supported in the Trial edition

Default project files

C++Builder stores your project information as follows:

- Form units are always saved as a unit of 3 files: .cpp, .h, .dfm (2 files for pascal based form units: .pas, .dfm)
- New units (first time opened) are saved as unit of 2 files: .cpp, .h
- Projects are saved as unit of 3 files: .mak, .cpp, .res
- You must save form units to a .cpp extension (.pas for pascal form units)
- You cannot save a new unit to a .h extension (any other ext is allowed)
- You must always save projects to a .mak extension

TFont::Charset property (additional information)

TFont See also

Charset specifies the character set of the font.

```
typedef Byte TFontCharset;
__property TFontCharset Charset;
```

Description

Set Charset to identify the character set of the font. Each typeface (specified by the Name property) supports one or more character sets. Check the information supplied by the font vendor to determine what values of Charset are valid.

The following table lists the predefined constants provided for standard character sets:

Constant	Value	Description
ANSI_CHARSET	0	ANSI characters.
DEFAULT_CHARSET	1	Font is chosen based solely on *Name* and *Size*. If the described font is not available on the system, Windows will substitute another font.
SYMBOL_CHARSET	2	Standard symbol set.
MAC_CHARSET	77	Macintosh characters. Not available on NT 3.51.
SHIFTJIS_CHARSET	128	Japanese shift-jis characters.
HANGEUL_CHARSET	129	Korean characters (Wansung).
JOHAB_CHARSET	130	Korean characters (Johab). Not available on NT 3.51
GB2312_CHARSET	134	Simplified Chinese characters (mainland china).
CHINESEBIG5_CHARSET	136	Traditional Chinese characters (Taiwanese).
GREEK_CHARSET	161	Greek characters. Not available on NT 3.51.
TURKISH_CHARSET	162	Turkish characters. Not available on NT 3.51
VIETNAMESE_CHARSET	163	Vietnamese characters. Not available on NT 3.51.
HEBREW_CHARSET	177	Hebrew characters. Not available on NT 3.51
ARABIC_CHARSET	178	Arabic characters. Not available on NT 3.51
BALTIC_CHARSET	186	Baltic characters. Not available on NT 3.51.
RUSSIAN_CHARSET	204	Cyrillic characters. Not available on NT 3.51.
THAI_CHARSET	222	Thai characters. Not available on NT 3.51
EASTEUROPE_CHARSET	238	Includes diacritical marks for eastern European countries. Not available on NT 3.51.
OEM_CHARSET	255	Depends on the codepage of the operating system.

TWinControl::ImeMode (additional information)

TWinControl See also

The ImeMode property specifies the input method editor (IME) mode for the control.

```
enum TImeMode { imDisable, imClose, imOpen, imDontCare, imSAlpha, imAlpha,
  imHira, imSKata, imKata, imChinese, imSHanguel, imHanguel };
__property TImeMode ImeMode;
```

Description

Set ImeMode to configure the way an IME processes user keystrokes. An IME is a front-end input processor for Asian language characters. The IME hooks all keyboard input, converts it to Asian characters in a conversion window, and sends the converted characters or strings on to the C++Builder application.

ImeMode allows a control to influence the type of conversion performed by the IME so that it is appropriate for the input expected by the control. For example, a control that only accepts numeric input might specify an ImeMode of imClose, as no conversion is necessary for numeric input.

ImeMode can have one of the following values:

Value	Meaning
imDisable	Shut down the IME. *imDisable* has no effect on Chinese, Taiwanese, or Korean IMEs.
imClose	Close the IME conversion window, but leave the IME running in the background. The IME can be re-activated by a hotkey combination.
imOpen	Open the IME conversion window. The conversion mode is the last conversion mode used by the IME.
imDontCare	Launch the IME if it is disabled. The conversion mode is the last conversion mode used by the IME.
imSAlpha	Open the IME conversion window and set the conversion mode to accept single-width Roman alphabet input.
imAlpha	Open the IME conversion window and set the conversion mode to accept double-width Roman alphabet input.
imHira	Open the IME conversion window and set the conversion mode to double-width Hiragana. *imHira* is only available for Japanese IMEs.
imSKata	Open the IME conversion window and set the conversion mode to single-width Katakana (Hankaku Katakana). *imSKata* is only available for Japanese IMEs.
imKata	Open the IME conversion window and set the conversion mode to double-width Katakana (Zenkaku Katakana). *imKata* is only available for Japanese IMEs.
imChinese	Open the IME conversion window and set the conversion mode to double-width Chinese. *imChinese* is only available for Chinese IMEs.
imSHanguel	Open the IME conversion window and set the conversion mode to single-width Hanguel. *imSHanguel* is only available for Korean IMEs.
imHanguel	Open the IME conversion window and set the conversion mode to double-width Hanguel. *imHanguel* is only available for Korean IMEs.

- The value of ImeMode only takes effect when the control receives focus. To change the value of ImeMode after the control has gotten input focus, call the SetIme method.

TWinControl::ImeName (additional information)

TWinControl See also

The ImeName property specifies the input method editor (IME) name for the control.

```
__property System::AnsiString ImeName;
```

Description

Set ImeName to specify which IME to use for converting keystrokes. An IME is a front-end input processor for Asian language characters. The IME hooks all keyboard input, converts it to Asian characters in a conversion window, and sends the converted characters or strings on to the C++Builder application.

ImeName must specify one of the IMEs that has been installed through the Windows control panel. The property inspector provides a drop-down list of all currently installed IMEs on the system. At runtime, applications can obtain a list of currently installed IMEs from the global Screen variable.

If ImeName specifies an unavailable IME, the IME that was active when the application started is used instead. No exception is generated.

- The value of ImeName only takes effect when the control receives focus. To change the value of ImeName after the control has gotten input focus, call the SetIme method.

Enabling UNICODE in your C++Builder application

Use the following procedures to UNICODE-enable your C++Builder console or Windows application.

- Make the following changes to your project source file (such as Project1.CPP):

1. Insert the following line after the line containing `#pragma stop`:

```
#include <tchar.h>
```

2. Add the prefix `_t` in front of

```
main (for example, _tmain)
```

or

```
WinMain (for example, _tWinMain)
```

3. Change the WinMain parameter list so it reads:

```
_tWinMain(HINSTANCE, HINSTANCE, LPTSTR, int)
```

- Make the following changes to your project make file (such as Project1.MAK):

1. In the line containing `CFLAG1=` , add **`-WU`** as the first switch.
2. In the line containing `ALLOBJ`, add `w` to the end of your startup code name (such as C0xxxw.OBJ).

Pascal interface procedure support in C++Builder

Borland C++Builder can support Pascal interface procedures which take HWND parameters only when compiled without the WINDOWS.H 'STRICT' macro defined. To do so, modify the Pascal source and change the HWND parameter type into a Pascal 'Pointer'.

In the body of the procedure, the parameter should be cast to the desired HWND type.

Due to the interaction of C++ name mangling, the WINDOWS.H macros, and the use of namespaces, it is not possible to use Pascal interface procedures with HWND types with the WINDOWS.H macro 'STRICT' defined.

For example:

```
unit HWND_User;
interface
  uses Windows;

procedure ExampleHWND(x : Pointer);  { x is really an HWND }

implementation

procedure TrueHWND(x : HWND);
begin
end;

procedure ExampleHWND(x : Pointer);
var
  h : HWND;
begin
  h := HWND(x);
  TrueHWND(h);
end;
end.
```

The following "STRICT" handles cannot be exported from Pascal and should be treated as the ExampleHWND above:

HACCEL
HBITMAP
HBRUSH
HCOLORSPACE
HDC
HDESK
HENHMETAFILE
HFONT
HGDIOBJ
HGLRC
HHOOK
HICON
HINST
HKEY
HKL
HMENU
HMETAFILE
HMODULE

HPALETTE
HPEN
HRGN
HRSRC
HSTR
HTASK
HWINSTA
HWND

Distributing applications not supported in the Trial edition

The Trial edition is an introductory version of Borland C++Builder and is not intended to be used to create applications that you wish to deploy and distribite. To distribute your C++Builder application, you should purchase one of the following full-featured versions:

- C++Builder Client/Server Suite
- C++Builder Professional edition
- C++Builder Standard edition

For more information, see Contacting Borland.