

BORLAND C++BUILDER 3

The most advanced C++ compiler and development  
environment ever created

## New Feature Demonstration Script

Borland International  
December 31, 2024

PRODUCTIVITY OF RAPID APPLICATION DEVELOPMENT – WRITE C++ APPLICATIONS IN A FRACTION OF THE TIME	3
USE EXISTING C AND C++ CODE – REUSE OBJECTS FOR FASTER INTEGRATION AND DEVELOPMENT	4
GET TO THE WEB. EASY INTERNET DEVELOPMENT AND DEPLOYMENT	4
C++BUILDER’S WEB BROKER GETS DATA TO THE WEB EASILY: SERVER SIDE WEB DEVELOPMENT HOW TO CREATE AN ACTIVEX CONTROL USING ACTIVEFORMS DECISION CUBE FOR MULTI-DIMENSIONAL ANALYSIS TO MAKE HELP MAKE BETTER DECISIONS, FASTER. CREATING A COMPONENT TEMPLATE FOR CODE REUSABILITY WEB DEPLOY USING ACTIVE FORMS AND COMPONENT TEMPLATES REDUCES DISTRIBUTION COSTS ACTIVEFORMS: WEB DEPLOYMENT SIMPLIFIES CODE DISTRIBUTION AND MAINTENANCE	9
DEBUG QUICKLY AND EFFICIENTLY - GET ROBUST APPLICATIONS TO PRODUCTION	9
DLL DEBUGGING NEW TOOLTIP EXPRESSION EVALUATION DEBUG INSPECTOR NEW LOCAL VARIABLE INSPECTION NEW MODULE VIEW CPU VIEW NEW EVENT LOG	
EASILY CREATE ONE-STEP ACTIVEX/ATL CONTROLS FOR COM REUSABILITY	12
HOW TO CREATE AN ACTIVEX CONTROL FROM AN EXISTING C++BUILDER CONTROL	
HOW TO CREATE A C++BUILDER CONTROL AND TURN IT INTO AN ACTIVEX CONTROL	12
HOW TO DEBUG AN ACTIVEX CONTROL HOW TO CREATE A PROPERTY PAGE FOR YOUR ACTIVEX CONTROL THE FLEXIBILITY OF CODE INSIGHT <b>Code Templates Wizard</b>	
HOW TO BUILD A MULTI-TIER DATABASE APPLICATION WITH REMOTE DATA BROKER AND CONSTRAINT BROKER	16
C++BUILDER 3: CLIENT AND SERVER TOOLS FOR SIMPLIFIED CLIENT/SERVER DEVELOPMENT: <b>SQL Explorer and the Constraint Broker</b> FINISHING THE COM SERVER AND IMPLEMENTING THE CONSTRAINT BROKER FOR DATA VALIDATION CODELESSLY CREATE THE CLIENT SIDE FOR THE MULTI-TIER APPLICATION BRIEFCASE, LOCALIZED SORTING AND UPDATES CREATES NEW AND POWERFUL TYPES OF APPLICATIONS PARTIAL DATA PACKETS	
VISUALIZE DATA WITH INTEGRATED REPORTING AND CHARTING	20
STARTING FROM THE DATABASE FORM WIZARD CREATE SCALABLE DATABASE APPLICATIONS ON TIME AND ON BUDGET	
VISUALIZE DATA WITH INTEGRATED REPORTING AND CHARTING	22
STARTING FROM A NEW APPLICATION SEAMLESSLY INTEGRATED DATABASE DEVELOPMENT	

TITLE:

**Productivity of Rapid Application Development – write C++ applications in a fraction of the time**

Summary:

Features:

1. File | New Application
2. From the Standard component page put two Buttons, two Radio Buttons, and an Edit box on the form
3. From the Win32 component page put an Animate control on the form
4. Select Form1 and change the property  
Caption = Idle
5. Select the Button1 component and change the property  
Caption = Start
6. Select the Button2 component and change the property  
Caption = Stop
7. Select the RadioButton1 component and change the properties  
Caption = Copy  
Checked = True
8. Select the RadioButton2 component and change the property  
Caption = Recycle
9. Select the Button2 component and change the property  
Caption = Stop
10. Select the Edit1 component and change the property  
Text = <your name>
11. Select the Animate1 component and change the property  
CommonAVI = aviCopyFile
12. Double click on Button1 then write this code:

```
void __fastcall TForm1::Button1Click(TObject *Sender)
{
    RadioButton1->Enabled = False;
    RadioButton2->Enabled = False;
    Animate1->Active=True;
    if (RadioButton1->Checked) {
        Form1->Caption="Copying "+Edit1->Text;
    }
    else {
        Form1->Caption="Recycling "+Edit1->Text;
    }
}
```
13. Double click on Button2 then write this code:

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
    Animate1->Active=False;
    RadioButton1->Enabled = True;
    RadioButton2->Enabled = True;
    Form1->Caption = "Idle";
}
```
14. Double click on RadioButton2 then write this code:

```
void __fastcall TForm1::RadioButton1Click(TObject *Sender)
{
    Animate1->CommonAVI = aviCopyFile;
}
```
15. Double click on RadioButton2 then write this code:

```
void __fastcall TForm1::RadioButton2Click(TObject *Sender)
{
    Animat1->CommonAVI = aviRecycleFile;
}
```

16. Run the app (F9)

**Title:**

**Use Exiting C and C++ code –  
Reuse objects for faster integration and development**

**Summary:**

**Features:**

**Steps:**

1.

**Comments:**

**Title:**

**Get to the Web. Easy Internet Development and Deployment**

**Summary:**

With C++Builder 3, you create high-speed, high-throughput web-delivered data applications using your existing client/server knowledge. C++Builder simplifies the development of Internet and Intranet applications in a variety of ways. You choose the best way to communicate information over the web and C++Builder creates HTML applications, Active Forms, and server objects that run in distributed, thin client environments.

**Features:**

ActiveForms	Web Deployment	Internet Explorer
ISAPI, NSAPI	Integrated Internet Components	HTML WebBroker

**Steps:**

1. Click on the Internet Tab Page
2. Slowly run the mouse over the components to view the selection of Internet Components.

**Comments:**

The integrated Internet component page contains over 25 components to build dedicated applications like mail servers, event dispatching and Internet newsgroups. The complete set of Internet components supports Internet standards like HTML, HTTP, FTP and POP – the standards you use in development.

**C++Builder's Web Broker gets data to the web easily: Server Side Web Development**

**Summary:**

C++Builder's open architecture enables high-productivity development of server side DLLs that bind directly to ISAPI and NSAPI. C++Builder's WebBroker builds HTML applications that can leverage Borland's Midas technology to create distributed, high speed, transaction processing, thin client applications. The resulting Internet and Intranet systems support large numbers of clients and large quantities of data.

**Steps:**

1. File | New | WebServer Application, click OK
2. Select ISAP/NSAPI DLL then click OK
3. From the Data Access Tab, put a Database component in the WebModule.
4. Select the Database Component and change the properties in the Object Inspector
 

Alias =	BCDemos
HandleShared =	True
DatabaseName =	MyWebData

5. From the Data Access Tab, put a Session component in the WebModule.
6. Select the Session Component and change the properties in the Object Inspector  
AutoSessionName = True

**Comments:** This sets up one instance of the BDE to handle all requests, this saves server resources and works with the database object to handle passwords and database access. Handleshared synchronizes handles to cursors. Once autosession is set to true, both the database and associated query objects automatically use the available session management.

7. From the Data Access Tab, put a Query component in the WebModule.
8. Select the Query component selected, change the properties in the Object Inspector  
DatabaseName = BCDEMOS
9. With the Query Component selected. In the Object Inspector, click on the SQL property to open the SQL Window, type the following SQL Statement:  
Select Company, City State, Phone from Customer  
Click OK to close the SQL Window.
10. With the Query component selected, change the properties in the Object Inspector  
Active = True
11. Double click the Query component to open the Query Fields Editor
12. Right click on the Query Fields Editor, select Add Fields. Highlight all the fields then click OK.
13. From the Internet Tab, put an HTMLQueryTableProducer component in the WebModule.
14. With the HTMLQueryTableProducer component selected, change the properties in the Object Inspector  
Query = Query1
15. Double Click HTML QueryTableProducer to open the HTML Editor
16. Edit the following properties in the HTML Editor  
BGColor = Yellow  
Border = 2

**TITLE:**

**How To Create an ActiveX Control using ActiveForms**

**Summary:**

ActiveForms are ActiveX Controls that use the C++Builder form as a container for other C++Builder components. ActiveForms can also publish ActiveX property pages and type libraries for adding high-speed functionality to other development environments, for example Internet Explorer, Visual Basic, Optima or PowerBuilder.

**STEPS:**

1. File | Close All
2. File | New | ActiveX page | ActiveForm, Click OK
3. Project | Options | Packages Tab  
uncheck Build With RunTime Packages
4. Change Tabs to Linker  
uncheck Dynamic RTL, Click OK
5. Click OK to accept the Defaults
6. On the Form place a TButton from the Standard Component Page
7. Double Click the Button on the form to enter code on the OnClick event
8. Enter this Code:  

```
WinExec("Calc", SW_SHOW);
```

  
to statically define the button action versus using a design time property
9. File | SaveAll  
Save ActiveFormImpl1 as: BtImpl

Save ActiveFormProj as BtProj

10. Project | Compile
11. Run | Register ActiveX Server
12. Go to VB and use this control (BTProj.OCX)

**Title:**

**Decision Cube for Multi-Dimensional Analysis to make help make better decisions, faster.**

**Summary:**

C++Builder 3 provides the fastest way to turn enterprise-wide data into actionable information. Graphically display enterprise or Internet data. Special features include the Decision Cube XTab for multi-dimensional data analysis, TeeCharts for flexible high-speed graphing of Decision Cube and database information, and Quick Reports which visually and easily designs reports. The idea is to put your business at the center of development not development at the center of business.

**Steps:**

1. File | Close All
2. File | New Application
3. From the VCL Decision Cube ComponentTab Page Place on the form
4. Drop:  
DecisionQuery
5. Right mouse click to call up DecisionQuery Editor  
From the DropDown ListBoxes set  
Database = BCDemos  
Table = Orders
6. Select the following available fields and move them to the Dimensions list box:  
SaleDate, ShipVia, Terms, PaymentMethod,
7. Move these fields to the summaries list box  
AmountPaid, click Add then select Sum  
ItemsTotal, click Add then select Count.
8. Click OK to finish the Decision Cube Query
9. From the DecisionCube Component Palette page drop  
Decision Cube, Decision Source, DecisionPivot, DecisionGrid
10. Set the following properties in the Object Inspector  
DecisionCube1->Dataset = DecisionQuery1  
DecisionSource1->DecisionCube = DecisionCube1  
DecisionPivot1->Align = alTop  
DecisionPivot1->DecisionSource = DecisionSource1  
DecisionGrid1->DecisionSource = DecisionSource1  
DecisionGrid1->Align = alClient  
DecisionQuery1->Active = True  
This connects the components together and activates the query to see the information at design time.
11. Choose to do any of the following
  - a) Deactivate or activate a dimension:  
Click on the DecisionPivot button (ie PaymentMethod )
  - b) Collapse a dimension in the Grid  
Click on the (-) sign in the dimension title
  - c) Expand a dimension in the Grid  
Click on the (+) sign in the dimension title
  - e) Pivot (exchange) two dimensions in the grid

- Drag ShipVia dimension to the Terms dimension
- f) Drill into a value
  - Right Click on DHL value and select drill in.
- g) Change the Display Value in the Grid
  - Click on the Sum Of AmtPaid button on the Pivot and choose an alternative value like Count of ItemsTotal
- 12. Select the Grid and turn the align property in the Object Inspector to alTop
- 13. Move the bottom of the grid up
- 14. From the Additional Palette page drop a splitter bar
  - Splitter1.align = alTop;
  - Splitter1.Cursor = crVSplit;
- 15. Drop a DecisionGraph onto the form from the DecisionCube Component Palette
  - DecisionGraph1.align = alClient
  - DecisionGraph1->DecisionSource = DecisionSource1
  - Automatically graphs the dimensions of the Cube without Code
- 16. Run the app (F9)

**Title:**

**Creating a Component Template for code reusability**

**Steps:**

1. With the Existing Decision Cube Application do the following
2. Edit | Select all to select all the components on the form
3. Component | Create Component Template
  - Palette Page: type in <YourName> then press OK
  - This creates a new reusable component which we will put into an ActiveForm
4. File | Close All

**TITLE:**

**Web Deploy using Active Forms and Component Templates reduces distribution costs**

**Summary:**

C++Builder 3 uses the ActiveX architecture to easily deploy thin-client, zero-configuration applications over the web thereby reducing the cost of software configuration and distribution. Web Deployment uses Microsoft standard application delivery mechanisms (CAB File delivery) and the web infrastructure to distribute your applications to a growing number of users on the Internet, Intranet, and throughout the enterprise.

**ActiveForms:**

1. File | New | ActiveX Page | ActiveForm
2. Click OK then Click OK again to accept the defaults
3. Make the Form Larger
4. Go to the <YourName> Component Tab Page and drop the Component you created in the Creating a Component Template Section. You have just created a new ActiveX out of the Decision Cube Components.

**Web Deployment simplifies code distribution and maintenance**

5. Create a new Directory on your system called Temp
6. Project | Web Deployment Options and fill in the dialog box as follows.
  - Target Dir: c:\temp
  - Target URL: file:///c:/temp
  - HTML Dir: c:\temp
 Then Click OK

This moved all the necessary files for a Web Deployed Application to the proper location. If this were on a WebServer, these would have most likely been on the server machine. Options include security with Code Signing, Delivering a

packaged application and delivering any additional files.

7. Project | Compile
8. Project | Web Deploy
9. Start Internet Explorer 3.x or 4.x
10. In the Internet Explorer enter the Address c:\temp\ActiveFormProj.HTM.  
The ActiveForm will automatically be downloaded to your machine, configured, and run from within the Browser.
11. You can make the ActiveForm fit your screen better by changing the HTML code at View | Source Code. To see the changes, press REFRESH.  
A full DecisionCube with Integrated TeeChart Graph will be running in IE3 or IE4
12. Close Internet Explorer <Alt><F4>



**Title:**

**Debug quickly and efficiently - get robust applications to production**

**Summary:**

**Features:**

Debug Inspector	Watches	Breakpoints
Module View	DLL Debugging	Event Logs
CPU View	ToolTip Expression Evaluation	Local Variable Inspection

### **DLL Debugging**

**Summary:**

**Steps:**

13. File | Close All
13. View | Project Manager (CTRL | ALT | F11)
  - Click NEW in Project Manager to create a new project
13. Select Application in New Items dialog and click OK
13. View | Project Manager (CTRL | ALT | F11)
  - Right Click on ProjectGroup1
  - Select Add New Project
13. Select DLL in New Items Dialog
13. Press OK
13. File | Save As | MyDLL.bpr
13. Move to the end of MyDLL.cpp
  - Press CtrlJ
  - Double click on Function Declaration
  - Using the inserted code template, enter the following code:

```
void __declspec(dllexport) DllFunc(char* string)
{
    ShowMessage(string);
}
```

13. View | Project Manager (CTRL | ALT | F11)
13. Select MyDll.cpp, right-click and select Build
13. In Project Manager, select Project1, right-click
  - Select Add
  - Drop down “Files of Type”, and change to \*.LIB
  - Select MyDll.lib
  - Click Open
13. In Project Manager
  - Click +Project1 and verify that it contains MyDll.lib
  - Click +Unit1
  - Select Form1 and Double click to open
13. From the Standard Page drop a Button on Form1
13. Double Click the Button to enter Unit1.cpp, Button1.Onclick . Enter the following:

```
char s[] = "I am in the DLL now";
DllFunc(s);
```
13. Predeclare your DLL's function in Unit1.cpp
  - Move to the top of Unit1.cpp
  - Place the cursor after the following two lines:

```
#pragma resource "*.dfm"
TForm1 *Form1;
```
  - On a new line, enter the following code:

```
void __declspec(dllimport) DllFunc(char*);
```
13. View | Project Manager
13. Project | Build All Projects
  - Double click **Project1** to make active (it will bold)
13. Press Button1 to test message
13. Return to MyDLL.cpp and double click in the margin to place a breakpoint on the

following line of code:  
`ShowMessage(string);`

### **New Tooltip Expression Evaluation**

13. Press Button1 in your running application, your application will move into debug mode and to take you to the breakpoint in.
13. Move your cursor over "string" and notice ToolTip expression evaluation of the variable's value.

### **Debug Inspector**

13. Doubleclick on "string"
  - Right Click, select Inspect (Alt F5)
13. Optional: Click on Unit1.Cpp Tab
  - Double Click on Form1
  - Right Click, select Inspect (Alt F5)

### **New Local Variable Inspection**

13. Select Run | Inspect Local Variables.
  - Click on the Unit1.cpp tab
  - Place a breakpoint next to `char s[] = "I am in the DLL now";`
  - Click the run button
  - Click Button1
  - Run | Inspect Local Variables

### **New Module View**

13. Select View | Modules to view all modules loaded to support process and their entry points.
13. Select MyDll.dll and view the exported functions contained within it.  
The upper-left pane shows each module name and the address at which it is loaded.  
The lower left pane shows a tree view display of source files used to build the module.  
The right pane shows a list of entry points into the module.

### **CPU View**

13. Select View | Cpu and view programs assembly instructions, cpu registers, cpu flags and memory.

### **New Event Log**

13. Select View | Event log and view the order of loaded modules and process starts and exits.

### **Note:**

This debugging example may also be created with MessageBox. Here is the code:

In MyDLL.cpp:

```
void __declspec(dllexport) DllFunc(HWND handle, AnsiString string)
{
    MessageBox(handle, string.c_str(), "", MB_OK);
}
```

In Unit1.cpp:

```
void __declspec(dllimport) DllFunc(HWND, AnsiString);
```

Also in Unit1.cpp

```
AnsiString s = "I am in the DLL now";
DllFunc(s);
```

TITLE:

## Easily create One-Step ActiveX/ATL controls for COM reusability

Summary:

### How To Create an ActiveX Control from an Existing C++Builder Control

From Design through Deployment, C++Builder 3 is the easiest object oriented development environment for the Enterprise. It is the most productive application environment for creating high speed, industry standard ActiveX components and COM objects. C++Builder ActiveX controls require no runtime interpreter and no extra distributable files. Professional developers should not compromise - high speed, easily distribution, and native code compilation.

Also, there are well over 1000 Third Party controls written natively for C++Builder that are ready to be used as ActiveX controls by using One-Step ActiveX with C++Builder 3.

Steps:

1. File | Close All -- File | New Application --> don't save
2. File | New | ActiveX Tab
3. Choose ActiveX Control, click OK
4. In the Dialog, set ClassName = TCCalendar from the DropDown
5. Click OK with all the defaults
6. Project | Compile
7. Run | Register ActiveX Server
8. Start VBCCE 5.0 or VB5
9. Create a New Exe
10. Right Mouse click on the Components bar and chose components
11. Select / check the CalendarXControl and press ok
12. Place the CalendarXControl on the VB Form

TITLE:

## How To Create a C++Builder Control and turn it into an ActiveX Control

Summary:

Easily create ActiveX Controls that can reuse these objects through-out the enterprise. Creating an ActiveX control by starting with the C++Builder VCL gives the developer the full power of Object Oriented methodologies, including: Polymorphism, Encapsulation and Inheritance, currently unavailable in VB's object-based paradigm.

Steps:

1. File | Close All
2. Component | New Component
3. Fill in new Component Expert with the DropDown listboxes  
Ancestor Type = TButton  
ClassName = TSoundPlayer
4. Click Install Button
5. Click OK at the next dialog to install with the default settings
6. Click No -- Do Not Rebuild Package
7. Right Click View Header (SoundPlayer.h)  
Type the asterisked {\*} lines into the appropriate places. (do not type the {})  

```
class PACKAGE TSoundPlayer : public TButton
{
private:
{*}  AnsiString FFileName;
__published:
{*} void __fastcall Click();
{*} __property AnsiString FileName = {read =
    FFileName,write = FFileName};
};
```
8. Right Mouse click to close page and save the file.

9. In SoundPlayer.cpp enter the {\*} lines:

```
#include <vcl.h>
#pragma hdrstop

{*} #include <mmsystem.h>
#include "SoundPlayer.h"
#pragma package (smart_init)

static inline void ValidCtrCheck(TSoundPlayer *)
{
    new TSoundPlayer(NULL);
}

__fastcall TSoundPlayer::TSoundPlayer(TComponent* Owner)
: TButton(Owner)
{
}

{*} void __fastcall TSoundPlayer::Click()
{*} {
{*} if ((FFileName != "") && (FileExists(FFileName)))
{*}     PlaySound(FFileName.c_str(), 0, SND_ASYNC);
{*} }
```

10. Project | Build All Projects

11. Component | Install Packages

In the Install Package Wizard

Click the Add Button

Select DCLUSR35.BPL

Hit OK button

Move the DCLUSR35.BPL file from the Lib directory to the Bin Directory

12. File|New Application

Say yes to save changes to DCLUSR35.BPL

Click the Save Button to save SoundPlayer.cpp

13. Drop the TSoundPlayer on the new project form from the Samples Component

Pallette

14. In the Object Inspector set the FileName property to any WAV file

for example: c:\windows\Update.wav

15. Run the app (F9) and click SoundPlayer1 --> The sound will play

16. File | Close All --> Don't Save any open files.

17. File | New | ActiveX Tab Page

18. Choose ActiveX Control

ClassName = TSoundPlayer

19. Click OK with all the defaults

20. Project | Make

21. Run| Register ActiveX Server

22. Start VBCCE 5.0 or VB5

23. Create a New Exe

24. Right Mouse click on the Components bar and chose components...

25. Select / check the SoundPlayerXControl and press ok

26. Place the SoundPlayerXControl on the VB Form

27. Change the Command Property to any WAV File

28. Run the project and click the button --> the WAV file will play

TITLE:

**How To Debug an ActiveX Control**

Summary:

With C++Builder's new DLL Debugging, it is now very easy to debug ActiveX controls

**Steps:**

from within another application environment.

26. Close VB
27. Place a break point in the SoundPlayerImpl file's STDMETHODIMP  
TSoundPlayerXImpl::Click() Section in the Implementation on the line

```
m_VclCtl->Click();
```

28. RUN | Parameters  
Host Application is VB5 CCE executable, wherever that may be on your disk
29. Press the Run Button (F9)
30. Go Back to Line 22

### How To Create a Property Page for your ActiveX control

**Summary:**

C++Builder 3 has complete implementation of ActiveX and COM. This means creating property pages in C++Builder is just as easy as creating the control.

**Steps:**

31. Close VB
32. With the SoundPlayer Project Still Open...
33. File | New | ActiveX Page | Property Page
34. Put a Static Text Component on the form from the Additional VCL Component Tab
35. Put an Edit Box on the form from the Standard VCL Component Tab
36. Change the Following Properties in the Object Inspector

```
Form->Name = PropPage
Edit1->Text = <blank>
StaticText1->Caption = FileName
```

37. File | Save All

```
Name Unit1 to PP1
Press OK for the other files to accept defaults
```

38. In the PP1.cpp file put the code marked //new in the procedures listed below --> the procedure is already stubbed.

```
void __fastcall TPropertyPage1::UpdatePropertyPage(void)
{
    Edit1->Text = OleObject.OlePropertyGet("FileName"); //New
}
```

```
void __fastcall TPropertyPage1::UpdateObject(void)
{
    OleObject.OlePropertySet("FileName",
        Variant(Edit1->Text)); //New
};
```

39. In the SoundPlayerImpl header file, include after the #include "soundplayer.h" entry the following:

```
#include "pp1.h"
```

40. In the SoundPlayerImpl header file put the code marked //new in the procedure listed below. The procedure is already stubbed.

```
BEGIN_PROPERTY_MAP(TSoundPlayerXImpl)
    // PROP_PAGE(CLSID_SoundPlayerXPage)
    PROP_PAGE(CLSID_PropertyPage1) //new
END_PROPERTY_MAP()
```

41. Press the Run Button (F9)

42. Go Back to Line 22 and stop at line 27

**Title:****The Flexibility of Code Insight****Summary:****Code Templates Wizard**

1. In the Button1ClickEvent - Type:  
`If <Ctrl-J>`  
to see a drop down list of pre-defined syntax for Delphi constructs
2. Select IFB shortcut from the list to have the code automatically fill with proper syntax thus preventing coding errors
3. Delete the inserted lines
4. Type IFB and <ctrl-J> and the code will automatically fill in because C++Builder realized you typed a shortcut.
5. Delete the inserted lines
6. On the menu Tools | Environment Options | Code Insight Tab
7. Click the Add button for Code Templates and enter  
`ShortCut: DLLE`  
`Description: DLLEexport`
8. Click OK
9. In the Coding section type:  
`void __declspec(dllexport) | //this is a pipe char`  
`{`  
`}`
10. Click OK button
11. In the Code Editor Type: DLLE <Ctrl-J>
12. Delete the inserted lines

**Title:****How to Build a Multi-Tier Database application with Remote Data Broker and Constraint Broker****Summary:**

C++Builder 3's Remote Data Broker is an essential technology that allows for an ultra-thin client architecture with the lowest application configuration and distribution costs. Using OLEnterprise or Distributed COM and the Remote Data Broker, the developer can efficiently communicate data from an application server to a client in a multi-tier application environment. C++Builder 3's unique Remote Data Broker reduces network traffic, centralizes transaction management, security, and performance, reduces network loads, and reduces the number of simultaneous database connections.

**Steps:**

1. File | New Application
2. File | New | Remote Data Module, Click OK and fill in Dialog as follows:  
`ClassName: DataBroker`  
Click OK
3. Make Form1 smaller and off center. In the Object Inspector  
Form1->Caption = Remote Data Broker
4. From the DataAccess Component Palette drop onto the DataBroker Remote Data Module, the following components:  
Query  
Session  
DataBase

These components are the basis of a multi-threaded database COM object server in a multi-tier environment.

TITLE:

**C++Builder 3: Client and Server tools for simplified Client/Server development:**

**SQL Explorer and the Constraint Broker**

**Summary:**

C++Builder 3 Client / Server Suite includes an integrated suite of tools for building high performance Client / Server applications with such extensive features as:

- Object Oriented Database Application Architecture
- Flexible Client / Server transaction models.
- Centralized Object Repository for team development
- SQL Monitor for SQL Testing and Tuning
- SQL Explorer for integrated administration of database servers
- Data Module and Remote Data Module Objects for separating business logic from visual data representation
- Scaleable Database Dictionary for consistent use of extended field attributes

The **Data Module**, **Scaleable Database Dictionary** and **Object Repository** are technologies that become the framework of a unique **Database Application Architecture**. This architecture then becomes the basis for building high performance Client / Server applications.

**Steps**

5. Select Database | Explorer from the Menu
6. Right Mouse Click on the "Database" Node in the Databases Tab
7. Select New <Ctrl-N> and supply the drivename from the ensuing dialog box then click OK  
Database Driver Name = INTRBASE
8. Type IBEmployee for the name (you will be in name edit mode already)
9. Make sure the ServerName property in the definition panel points to the example InterBase Database (this is the default location).  
e.g. C:\Program Files\Borland\IntrBase\EXAMPLES\employee.gdb
10. Change the UserName from MyName to sysdba
11. Post the Changes by hitting the Blue Arrow Speed Button or <Ctrl-A> and Press OK to save the changes
12. Double click and open the database IBEmployee and enter the password in the dialog box  
Password=masterkey  
Press OK
13. Right Mouse Click on the IBEmployee database and select Import To Dictionary  
Press OK  
This only has to be done once to import the constraints of the database to the dictionary. This may take a minute or so. You could just import a specific table by drilling down in the SQLExplorer tree view
14. Close the SQL Explorer <Alt> <F4>

**Finishing the COM server and implementing the Constraint Broker for data validation**

1. Using the Object Inspector change the properties of the following components. These components are in the DataBroker Window
2. Database1:  
AliasName: IBEmployee



- DatabaseName: IBEmployee
  - HandleShared: True
  - LoginPrompt: False
  - Double Click on Params and enter
    - UserName=sysdba
    - Password=masterkey
  - Click OK
- 3. Session1:
  - AutoSessionName = True
- 4. Query1
  - Name: Department
  - Double Click the SQL property and enter:
    - Select \* from Department
  - Click OK
  - DatabaseName: IBEmployee
- 5. Set the Department query Active Property to True. Set back to false.
- 6. Right Mouse Click on the Department Query and select
  - Export Department from DataModule
- 7. Right Mouse Click on the Department Query and select Fields Editor
- 8. Right Mouse Click in the Fields Editor and select Add Fields then OK
 

This puts the constraints from the Relational Database, which have been imported into the dictionary, into the application. This can be updated at runtime with the UpdateDataSet function in the DRINTF unit.
- 9. File | Save All
  - Save Unit2 as: BrokerRDM.cpp
  - Save Unit1 as: BrokerForm.cpp
  - Save the Project as: Broker.bpr
- 10. Run | Run (F9) → Run the app to register the COM Server
- 11. Close Running App (Alt F4)
- 12. View | Project Manager
- 13. In the Project Manager, Right Click on the Project Group and Add Project
- 14. From the Object Repository choose the Application icon

### Codelessly Create the Client Side for the Multi-Tier Application

1. From the Data Access Component Page Drop the following components on the form
  - RemoteServer
  - ClientDataSet
  - DataSource
2. Change MIDASConnection1 Properties in the Object Inspector
  - ServerName: Broker.DataBroker from the DropDown List
  - Connected: True

If you were doing this on two machines you would move the remote server Broker.Exe to another machine. By running it on that machine you would register the COM server. Then in the client application you would set the Computer Name to the remote machine, thereby setting the DCOM server. (note you need DCOM on both machines and the InterBase Database Location may need to change)
3. Change ClientDataSet1 Properties in the Object Inspector
  - MidasConnection = RemoteServer1 from the DropDown List

ProviderName = Department from the DropDown List  
Active = True

4. Change the DataSource1 Property in the Object Inspector  
Dataset = ClientDataset1 from the DropDown List
5. Drop a DBGrid and a DBNavigator on the form from the DataControls Component Palette
6. Select both the DBGrid and DBNavigator (shift click each component) and change the DataSource Property in the object inspector  
DataSource = DataSource1  
to see the data from a remote datasource at DesignTime
7. Run | Run
8. Enter the Value 2 in the DBGrid for the Budget Field and try posting by clicking on the next row. The Constraint Broker won't allow the illegal value even though it never goes back through the network to the Relational Database. This reduces network traffic.

**Title:**

### **BriefCase, localized sorting and updates creates new and powerful types of applications**

1. Close the running application <Alt F4>
2. Drop three buttons from the Standard Component Palette
3. In the Object Inspector set the following Properties  
Button1->Caption = Save to Disk  
Button2->Caption = Load from Disk  
Button3->Caption = Apply Changes to Server
4. Double Click on Button1 and enter the following code in the Button1Click event  
`ClientDataSet1->SaveToFile('C:\Test.tst');`
5. Double Click on Button2 and enter the following code in the Button2Click event  
`ClientDataSet1->LoadFromFile('C:\Test.tst');`
6. Double Click on Button3 and enter the following code in the Button3Click event  
`ClientDataSet1->ApplyUpdates(0);`
7. Click on the DBGrid control and go to the Object Inspect Events Page
8. Double click on the OnTitleClick Event and insert the following code  
`ClientDataSet1->IndexFieldNames := Column->FieldName;`
9. Run the Application (F9) and you can persist data to disk, apply changes to the Remote Data Broker Server, reload data from Disk, and Sort the Data by clicking on the columns header in the grid. Perfect for sales automation type applications where people work off line for extended periods of time.

**Title:**

### **Partial Data Packets**

**Summary:**

Unique to the Remote DataBroker technology is the concept of the partial datapacket. This means that it is up to the design of the project and the programmer to determine whether it is necessary to download an entire result set from a query. This is critical design for efficient network usage and works particularly well with master-detail applications.

1. Close the running application <Alt F4>
2. Drop a StatusBar from the Win32 Component Palette onto the form

3. Change StatusBar1 properties in the Object Inspector  
SimplePanel = True
4. Click on DataSource1 and in the Object Inspector go to the events tab.
5. Double Click the OnDataChange event and enter the following code
 

```
char s[32];
wsprintf(s, "%d of %d",
ClientDataSet1->RecNo,ClientDataSet1->RecordCount);
StatusBar1->SimpleText = s;
```
6. Change the Properties in the ClientDataSet1 with the Object Inspector  
PacketRecords = 5  
(-1 is all, 0 is meta data only, number is quantity of records)
7. Run the Application (F9)  
As you scroll through the records in the grid, the StatusBar will indicate the records that have been pulled from the Remote Data Broker. If you request more records from the Remote Data Broker Server, the application will automatically do this and append it to the ClientDataset.

**Title:**

**Visualize Data with Integrated Reporting and Charting**

**Starting from the Database Form Wizard**

**Create Scalable Database Applications On Time and On Budget**

**Summary:**

**Features:**

1. File | Close All
2. File | New Application
3. DataBase | Form Wizard
4. Click on Master Detail Radio Button then click Next
5. From the Drive Alias or Name DropDown List Box choose BCBDEMOs
6. From the TableName list box choose Customer.db then click Next
7. Choose the ">>" button to move all the available fields to the selected fields list box then click Next button
8. Click Next button choosing to use the Horizontal setting
9. Choose the Detail table Orders.db from the TableName list box then click the Next Button
10. Choose the ">>" button to move all the available fields to the selected fields list box then click Next button
11. Hit the Next button to accept the default GRID selection
12. From Available indexes DropDown ListBox choose CustNo, in the detail and master fields listbox choose CustNo then click the Add Button and then the Next Button
13. Choose the Form and DataModule Radio Button then Click Finish Button
14. In the DataModule2 Form change table1 and table2 property settings:  
Active = True
15. Drop a Button from the Standard Palette next to the navigator bar in Form 3
16. Double Click the Button to enter the following code into the Click Event Handler

```
QuickReport4->Preview();
```

17. File | New
18. In the Object Inspector double click the Report icon
19. File | Include Unit Header and choose Unit2 to reference the reusable DataModule then click OK
20. In the Object Inspector
21. Click the QuickReport4 form and change the following properties in the Object Inspector

QuickReport4

Double Click on the Bands Property and set  
HasDetail = True  
DataSet = DataModule2->Table1

22. In QuickReport4 form drag the detail band down
23. From the QReport Component Pallete drop down a QRDBText and TQRChart component and set the following properties in the Object Inspector

QRDBText

DataSet = DataModule2->Table1  
DataField = Company

24. Double click the TQRChart Component to use the property editor  
Click the Add Button, Choose Line, then hit the OK button  
Click the Series Tab, then the DataSource Tab  
From the DropDown List Box choose DataSet  
From the DataSet DropDown List Box choose DataModule2->Table2  
Click the DateTime checkBox for the X axis  
For the Y axis choose the AmountPaid Field from the DropDown List Box  
Hit the Close Button

25. Right Mouse Click in the QuickReport4 Form and choose Preview

**Comment:**

You can preview a report without having to run the whole application. This is integrated and interactive reporting. Also, notice that the report engine is threaded so that you can look at data while it is compiling the rest of the report in the background.

26. GoBack to Form3
27. File | Include Unit Header Unit4
28. Project | Run
29. Click the button to preview the report

**Title:**

**Visualize Data with Integrated Reporting and Charting**  
**Starting From a New Application**  
**Seamlessly Integrated Database Development**

**Summary:**

**Features:**

**Steps:**

1. File | Close All
2. File | New Application

3. File | New
4. From the Object Repository choose DataModule
5. In the DataModule drop the following components:
  - 2 TTables
  - 2 DataSources
6. Set the following properties on each of the components
  - TTable1
    - Database = BCBDemos
    - TableName = Customer
    - Active = True
  - TDataSet1
    - DataSet = TTable1
  - TDataSet2
    - DataSet = TTable2
  - TTable2
    - Database = BCBDemos
    - TableName = Orders
    - MasterSource = DataSet1

To change the MasterFields property click on the elipsis in the object inspector change the available index to custno and select custno in the detail and master list boxes then click the add button and then the ok button

Active = True

7. Click on Form1
8. File | Include Unit Header ---From the dialog box choose Unit 2 then click OK
9. Drop Two DBGrids from the DataControls Page onto form1
10. Change the Properties of the respective DBGrid Controls as indicated
  - DBGrid1
    - DataSource = DataModule2->DataSource1
  - DBGrid2
    - DataSource = DataModule2->DataSource2

#### **Comment**

You have just created a very simple master detail application. You could easily have done this with the DataForm wizard

11. File | New
12. From the Object Repository click the Report icon and then OK
13. File | Include Unit Header --From the dialog box choose Unit 2 then click OK
14. Click the QuickReport3 form and change the following properties in the Object Inspector
  - QuickReport3
    - Double Click on the Bands Property and set
      - HasDetail = True
      - DataSet = DataModule2->Table1
15. In QuickReport3 form drag the detail band down
16. From the QReport Component Pallete drop down a QRDBText and TQRChart component and set the following properties in the Object Inspector

QRDBText

DataSet = DataModule2->Table1

DataField = Company

17. Double click the TQRChart Component to use the property editor  
Click the Add Button, Choose Line, then hit the OK button  
Click the Series Tab, then the DataSource Tab  
From the DropDown List Box choose DataSet  
From the DataSet DropDown List Box choose DataModule2->Table2  
Click the DateTime checkBox for the X axis  
For the Y axis choose the AmountPaid Field from the DropDown List Box  
Hit the Close Button

18. Right Mouse Click in the QuickReport3 Form and choose Preview

**Comment:**

You can preview a report without having to run the whole application. This is integrated and interactive reporting. Also, notice that the report engine is threaded so that you can look at data while it is compiling the rest of the report in the background.

19. GoBack to Form1
20. File | Include Unit Header 3
21. Drop a Button on the Form from the Standard Pallete Page
22. Double Click the Button and put the following code in the ButtonClick Event Handler  

```
QuickReport3->Preview();
```
23. Project | Run
24. Click the button to preview the report