What's new

Delphi 5 introduces the following new features and enhancements.

Note Some of the features are not available in all versions of Delphi.

- ADO Dataset (Enterprise and add-on to Professional edition)
  The ADO dataset provides an alternative technology to the Borland Database Engine (BDE) to gain access to data in a variety of formats using Microsoft's Active Data Objects (ADO) technology.
- Data Module Designer (All editions)
  The Data Module Designer is a new visual design tool that makes it easy to create and maintain data modules.
- InterBase Express (Professional and Enterprise editions)
  InterBase Express (IBX) components integrate InterBase with Delphi better than any other database access components and do not require the Borland Database Engine (BDE).
- MIDAS enhancements (Enterprise edition)
  The architecture to support multi-tier database (MIDAS) applications now supports stateless remote data modules and the new InternetExpress components let you create Web applications where browsers interact with data from a MIDAS application server.
- **CORBA changes** (Enterprise edition)
  CORBA has been upgraded to work with the VisiBroker for C++ ORB version 3.32. In addition, message traffic is reduced because the CORBA client no longer pings messages to the server to maintain the client connection. Connections time out automatically if they are inactive for too long.
- New debugging features (All editions)
  The Integrated debugger has new features, including the ability to set debugging options for specific processes, new Run menu items, and additional debugging options.
- VCL enhancements (All editions)
  The VCL has many new objects, properties, and events incorporating new technology and responding to developer's requests.
- Frames (All editions)
  Frames are a special kind of form that can be nested within a form or another frame. You create frames by choosing File|New Frame, or by choosing File|New and double-clicking Frame on the New tab of the New Items dialog box.
- Customizable desktop settings (All editions)
  You can customize various desktop layouts, then name and save them globally. Select the desktop layout you want from the Desktop toolbar or using View|Desktops. You can also specify a debugging desktop to load during runtime.
- To-Do Lists (Professional and Enterprise editions)
  A to-do list maintains a list of tasks to be completed for the project. You can add items in the source code or directly to the list.
- Property categories in the Object Inspector (All editions)
  The Object Inspector allows you to display and filter properties and events by category.
- Images in drop down lists in the Object Inspector (All editions)
  Owner draw support has been added to the Object Inspector so you can view images such as cursors, image lists, and colors in drop down lists.
- New Project Manager features (All editions)
  The Project Manager simplifies project management by allowing you to drag and drop files from Windows folders or other projects into your current project. You can also copy project items from one project to another and add any type of file to a project. Resource files that you add to your project are compiled into RES files and linked to the project.
- TeamSource (Enterprise edition)
  TeamSource is a new integrated workflow management tool. For more information, see TeamSrc.hlp in the Help directory. **Note:** *TeamSource is a separate product and a separate installation, and is not provided with all versions of Delphi. It can be purchased separately.*

- ActiveX Enhancements (Professional and Enterprise editions)
  Imported COM servers can be implemented as components to for visual development. Other features have been added to simplify ActiveX development.
- **New application wizards** (Professional and Enterprise editions)
  Two new wizards help with creating control panel applets (File|New then select Control Panel Application or Control Panel Module) and console applications (File|New then select Console Wizard).
- Editor enhancements (All editions)
  Editing preferences are now centralized under a separate command (Tools|Editor Options). It is now easier to customize editor key binding with the Key Bindings tab on the Editor Options and Open Tools API enhancements.
- New project browser (All editions)
  A new project browser can browse symbols in your project (project-wide) or it can browse all symbols (VCL-wide).
- **Forms now saved as text** (All editions)
  Form files (DFMs) are now saved as text rather than binary, by default. Right-click on a form and uncheck Text DFM to save that form as binary. Uncheck the New forms as text on the Preferences page of the Tools|Environment Options to save forms in binary.
- **Option to auto create forms** (All editions)
  An option on Tools|Environment Options Preferences page lets you choose whether or not to automatically create forms. You must have at least one auto-created form to use this option. When unchecked, forms added to the project are put into the Available Forms list rather than the Auto Create list. You can change where forms are listed using the Forms tab of the Project|Options dialog box.
- IDE command line options (All editions)
  You can start the IDE from the command line using several new options particularly useful for debugging.
- International tools (Enterprise edition)
  A new suite of tools called the Integrated Translation Environment (ITE) is now provided to simplify software localization and simultaneous development for different locales.
- **NetMasters components** (Professional and Enterprise editions)
  The NetMasters components now appear on a separate page (called FastNet) in the Component Palette. Note that the TWebBrowser component on the Internet page, which imports the Internet Explorer ActiveX component, replaces the THTML component.
- Help menu changes (All editions)

ADO Dataset **(Enterprise & add-on to Professional edition)**

Delphi 5 includes new components that you can use to access data through Microsoft's ActiveX Data Objects (ADO). ADO is Microsoft's high-level interface to all kinds of data. This application-level interface to Microsoft's data access technology is called OLE DB. OLE DB provides fast access to any data source, including relational and non-relational databases, email and file systems, text and graphics, and custom business objects.

Delphi's new components provide data access using ADO technology through existing data aware controls (such as DBGrid and DBEdit) without requiring the Borland Database Engine (BDE). The ADO/OLE-DB runtime must be installed to use these components.

The following components support ADO technology:

Component	Description
TADODataSet	A TDataSet descendant that uses the ADO RecordSet object to access data. It can be used as a replacement for any of the existing TTable, TQuery, or TStoredProc components.
TADOTable	Specialized TADODataSet which is used for accessing data in a table. It can be used in place of the existing TTable component.
TADOQuery	Specialized TADODataSet for replacement of the BDE TQuery component.
TADOStoredProc	Specialized TADODataSet for replacement of the BDE TStoredProc component.
TADOConnection	Provides a connection to an ADO database.
TADOCommand	Allows for specialized execution of SQL statements.

In addition, the following new field types have been added to provide access to the new data types supported in ADO:

- WideString
- GUID
- Variant
- Interface
- IDispatch

InterBase Express **(Professional and Enterprise editions)**

Delphi 5 includes new components that you can use to access data through InterBase Express (IBX). IBX applications work better, perform faster, and give you access to advanced features of InterBase never before available in standard components. IBX is designed to serve the needs of InterBase developers, and to provide the highest performance component interface for InterBase 5.5 and later.

IBX is based on the custom data access Delphi component architecture, and is integrated with the Delphi 5 Data Module Designer, so developers who are familiar with traditional data access components will feel right at home.

IBX is compatible with Delphi's rich library of data-aware components, and does not require the Borland Database Engine (BDE).

The following components support IBX technology:

Component	Description
TIBTable	Custom dataset which is used for accessing data in a table or a view. It can be used in place of the BDE TTable component.
TIBQuery	Custom dataset for replacement of the BDE TQuery component.
TIBStoredProc	Custom dataset used for executing stored procedures.
TIBDatabase	Provides a connection to an InterBase database.
TIBTransaction	Allows you to access all the powerful options of InterBase transactions. Your applications benefit in performance and concurrency by using the most appropriate transaction options for a given situation. You can maintain multiple concurrent transactions on one or more databases, to interleave data operations that need to remain logically atomic. TIBTransaction supports distributed transactions involving multiple databases, allowing you to implement integrity enforcement and data consistency across multiple databases.
TIBUpdateSQL	Defines custom actions for updating normally read-only tables and for caching updates on the client. Enables you to design normalized databases without restricting your ability to design applications to update complex datasets. Enables you to implement code by design, and not by working around your database implementation.
TIBSQL	Executes SQL statements and retrieves data at high speed without the overhead of buffering data and interfacing with Delphi data-aware controls. This is the most direct way to access InterBase data.
TIBDataSet	A custom dataset, similar to TIBQuery, that provides live access to InterBase data.
TIBDatabaseInfo	Enables applications to request information about a database or an InterBase server. This includes database properties, system performance data, users currently connected to the database, and other information. Developers can use this component interface to write performance-monitoring applications and database automation tools.
TIBSQLMonitor	Allows for advanced debugging of data communications, resulting in faster engineering for client/server and multi-tier InterBase projects.
TIBEvents	Provides a means for your application to register interest in, and asynchronously handle, events posted by an InterBase server.

MIDAS enhancements **(Enterprise edition)**

This section describes enhancements to Multi-tier Distributed Application Services Suite (MIDAS) as of Delphi 5 (MIDAS). See Understanding MIDAS technology for an overview of this technology and the architecture of a three-tiered application built using MIDAS. Refer also to Changes to MIDAS support for detailed information about architectural changes.

Stateless data broker

The architecture to support MIDAS applications has changed to support stateless remote data modules. This has important advantages over the previous architecture. You can now write MTS servers and shared remote data module instances without creating your own custom interfaces. The new architecture improves performance by reducing message traffic: Each call from the client application to the application server carries more information, so fewer calls are needed. New interfaces make it easy to transmit application-specific information whenever the client application calls the application server.

The architectural changes do, however, require some changes to existing applications if they are to work with the new MIDAS support. See Converting MIDAS applications for details.

InternetExpress applications

A new InternetExpress page has been added to the Component Palette that contains components for building internet applications. InternetExpress applications provide a way to distribute Midas-based multi-tiered applications to clients over the Internet.

To create InternetExpress applications, you need to replace the client tier with a special Web application that acts simultaneously as a client to the application server and as a Web server application that is installed with a Web server on the same machine. See Building HTML-based Web applications using InternetExpress for information on building applications that allow javascript-enabled browsers to interact with data from your application server through HTML pages. You may also want to see Writing MIDAS Web applications for a comparison of this approach with the ActiveX-based approach supported by previous versions of Delphi.

Constraints and defaults

Constraints and default values now work when placed on a TClientDataSet using either the TField properties or a TCheckConstraint.

Web connection component

A new connection component, TWebConnection, uses HTTP to establish a connection between the client and the application server. Because this component uses HTTP protocol, you can use it for connecting through a firewall or taking advantage of the SSL security provided by HTTPS.

Pooling

Delphi now allows for pooling of remote data modules.

Socket Server

The ScktSrvc.exe and ScktSrvr.exe have been combined into ScktSrvr.exe for this release, and it can be used as a service or not. The syntax is as follows:

Format	Description
`scktsrvr –install`	To install it as a service
`scktsrvr –uninstall`	To uninstall it
`scktsrvr`	To run the Socket Server as a normal application

If the service manager starts it, then it will start as a service.

TDataSetProvider

TDataSetProvider can now apply updates directly to the database server as well as to a dataset. Dataset components implement an interface that enables them to work with a dataset provider. This

allows TDataSetProvider to work with any kind of dataset (including BDE-enabled, InterBase Express, or ADO-based as well as client datasets or custom datasets). TDataSetProvider does not require the Borland Database Engine (BDE) or DBOLE. New applications should use TDataSetProvider instead of a TProvider component, which is retained for backward compatibility only.

New debugging features (All editions)

Delphi 5 has many new and improved debugging features, including:

New debugging view

- FPU window is a new IDE debugger window that lets you view the contents of the Floating-Point Unit in the CPU. The FPU window displays register values, status, control, and tag words. You can display floating-point or MMX information in the FPU window. (Not available in the Standard version of Delphi.)

Breakpoint features

- Breakpoint actions provide various things that breakpoints can do when they are encountered.
- Breakpoint groups let you organize breakpoints into groups that you can enable or disable all at once.
- Breakpoint properties, associated actions, and group names are displayed in a tooltip. Point the cursor at a breakpoint glyph in the gutter of the Code editor. A tooltip is displayed that shows the breakpoint's pass and condition.
- Breakpoint List includes additional columns to show the actions associated with each breakpoint and its group name (if any).

New debugging commands and options

- Run|Attach to Process command allows you to debug a process that's already running outside the IDE. Choosing this command displays a list of active processes. Select the one you want to attach to your debugging session.
- Run|Run Until Return command executes your program until the current routine returns to its caller. This is useful when you've accidentally single-stepped into a routine that you don't need to debug, or when you've determined that the current routine works to your satisfaction and you don't need to single-step through the rest of it.
- Allow function calls option for Watches when you select Run|Add Watch.
- Watch button on the Run|Evaluate/Modify dialog box so you can create a watch for the current expression.
- Inspect button on the Run|Evaluate/Modify dialog box so you can create a new inspector for the current expression.

Debugging usability enhancements

- Debugging now includes drag and drop support.
  While debugging, you can drag any expression from the editor to the Watch List (creates a watch for that expression), Debug Inspector window (inspects the contents of the expression), or the stack and dump panes in the CPU window (positions the pane to the address of the expression). When not debugging you must hold down the Alt key to drag and drop into the debugger views.
- Within the Watch list, you can right-click to display a Debug Inspector for the currently highlighted expression.
- In the Thread Status view, you can right-click on a process and select Process Properties to set temporary debugging options for that process.
- Within the Modules window, you can sort by entry point or address in the Entry Point pane.
- In the CPU window, a green arrow appears in the margin to note the direction of a call or jump.
- Many of the debugging windows now include multiselection of items and provide clipboard support.
- A new debugger event log option "Display process info with event" lets you choose whether or not to display the process name and OS process ID for the process that generated the event. It is on by default.
- Debugging info is now included with DCUs in the Debug directory.

VCL enhancements

The Delphi object hierarchy includes common control updates, ADO components, web enhancements, and many other new features. The following topics describe pertinent VCL changes.

- Common control updates
- Web enhancements
- Custom draw events
- Control panel applet wizard components
- Miscellaneous VCL enhancements
- VCL Documentation enhancements

Refer to Compatibility issues for VCL changes that could potentially impact applications being upgraded to Delphi 5.

Refer to MIDAS enhancements and Changes to MIDAS support for VCL changes that were implemented to support multi-tier database application development.

Common control updates **(All editions)**

Common control updates to the VCL include

- Custom draw support for TToolBar
- List view support for workareas (TListView)
- Subitem images in report-style list views (TListView)
- New HoverTime property lets users select list view items without clicking (TListView)
- InfoTip (also called Help Hint) support on all items in a list (TListView)
- A new OnColumnRightClick event (TListView).
- DragReorder for drag-drop reordering of header sections (THeaderControl)
- New OnSectionDrag event (THeaderControl).
- New ItemEnabled property that lets you programmatically enable or disable individual items in the list (TCheckListBox).

New "advanced" custom draw events were added for TTreeView, TListView, and TToolBar.

Web enhancements **(Professional and Enterprise editions)**

TWebBrowser, a new component, lets you embed a browser page in your application (using Internet Explorer 4.0 or later). At design-time, the Response Editor and the Web page editor of the TDataSetTableProducer and TQuerySetTableProducer components use TWebBrowser to let you preview your page in a browser.

Web server applications are easier to create with the new Producer property on TWebActionItem. This new property allows you to associate an HTML content producer with a Web action item so that it automatically updates the content of response messages when the action item is executed.

Page producers now have a StripParamQuotes property. This compensates for HTML editors that automatically insert quotation marks around the options added to a tag.

You can use packages when building an ISAPI/NSAPI DLL.

Note On the Enterprise edition, this change impacts existing WebBroker applications. See Upgrading WebBroker applications.

Control panel applet wizard components **(Professional and Enterprise editions)**

New objects, TAppletApplication and TAppletModule, support the Control Panel Applet wizard that makes it easy to design applets for the Windows control panel. EAppletException , a new exception class is also provided.

Custom draw events **(All editions)**

New "advanced" custom draw events were added for TTreeView, TListView, and TToolBar. These events have the same name as existing events except that the word Advanced is prepended. Thus: In addition to OnCustomDraw, there is also OnAdvancedCustomDraw; in addition to OnCustomDrawItem, there is also OnAdvancedCustomDrawItem; and so on.

The Advanced custom draw events differ from the existing events in that they occur more often (not just in the prepaint stage). The event handlers include an additional parameter that indicates the current state of the paint process (prepaint, postpaint, preerase, or posterase).

Miscellaneous VCL enhancements

InterBase Express (IBX) (Professional and Enterprise editions)
This new set of data access components provides a means of accessing data from InterBase databases.

ActiveX Data Objects (ADO) (Enterprise & add-on to Professional edition)
New components were added to support access to data through Microsoft's ActiveX Data Objects (ADO). Refer to ADO Dataset for more information.

Custom Constraints (Enterprise edition)
TClientDataSet now enforces custom constraints and defaults (using either the TField properties or a TCheckConstraint). The Constraints property is now published so you can save record-level constraints.

Database changes (Professional and Enterprise editions)
TDatabase now lets you execute an SQL statement without the overhead of using a TQuery object. You can use the public TDatabase.Execute method.

BDE-enabled datasets now support an AutoRefresh property. When you set AutoRefresh to True, default values and autoincrement values are fetched automatically when you post a record (without having to call Refresh).

New connection component (Enterprise edition)
TWebConnection, a new connection component, has been added. See MIDAS enhancements.

Keyboard layout properties (All editions)
TApplication has two new properties: BiDiKeyboard and NonBiDiKeyboard. These allow you to specify the keyboard layout.

Setting event handlers from the IDE (All editions)
TApplicationEvents is a new component that intercepts application-level events. Use this component as a way to set event handlers for application events using the IDE.

New property for splitters (All editions)
TSplitter has a new AutoSnap property that determines what happens when a user drags the splitter so as to resize a neighboring object smaller than the minimum size. When AutoSnap is True, such an attempt causes the neighboring object to be resized to zero height (or width). When AutoSnap is False, the resize stops at the minimum size.

New Help hint and menu features (All editions)
TScreen has two new properties: HintFont and MenuFont, which determine the font used for help hints and menu items, respectively.

TMenuItem.OnAdvancedDrawItem allows you to render owner-drawn menu items including state information (for example, checked, default, grayed). (OnDrawItem only gives you selected/not selected).

Component writers now have more control over context menus with the new PrepareItem method on TComponentEditor.

Menus and menu items now have an AutoHotKeys property. Setting AutoHotKeys to True causes the menu to maintain hot keys (also called accelerator keys), remove duplicate hot keys, and add unique hot keys to menu items that don't have them.

AutoLineReduction removes superfluous separator bars from menus. TMenuItem also includes several new features:

- Find a specific menu item (Find)
- Insert separator bars into a submenu (InsertNewLineAfter, InsertNewLineBefore, NewBottomLine, NewTopLine)

- Lets you store images with a submenu (SubMenuImages)

Invoking pop-up menus (All editions)

OnContextPopup is a new TControl property called in response to a WM_CONTEXTMENU message which, in turn, is called in response to a right-click (or keystroke invocation of the context menu). In this event, you can create and pop up your own menu or display a pop-up dialog instead of a menu.

TPopupMenu.MenuAnimation lets you configure the way the menu appears on Windows 2000. It can pop up or "slide" in like a window shade.

Page Control feature (All editions)

The Highlighted property of TTabSheet is a new property that lets you highlight pages in a page control. Multiple pages can be highlighted at the same time.

New container classes (All editions)

The new Contnrs unit introduces a number of utility classes for managing stacks and queues.

Decision cube source files (Enterprise edition)

The Delphi 5 product now includes the source files for the decision cube components in the \Source\ Decision Cube directory.

InternetExpress components (Professional and Enterprise editions)

An InternetExpress page has been added to the Component Palette. It provides components for building internet applications. See MIDAS enhancements for more information.

VCL documentation enhancements

The VCL documentation now includes topics describing the classes for each unit. In an object entry, click on the unit name for a list of all the objects in that unit.

Global routines also have separate topics both for units and categories.

Property categories in the Object Inspector **(All editions)**

You can now display and filter properties and events by category in the Object Inspector. By filtering the properties, you can reduce the number of properties visible in the Object Inspector and focus on those which are primarily of interest at the time. You can also more easily locate related properties by viewing them by category. For example, when localizing your application for other countries, you can display only properties that need to be localized by unchecking all categories except Localizable.

Component writers can create categories and assign properties to categories using the RegisterPropertyInCategory procedure. See Property categories: functions and classes and Property categories for details.

Filtering properties

To change the filter, right-click, choose View, and check or uncheck categories. Properties associated with checked categories are visible in the Object Inspector.

Displaying properties by category

To display properties by category, right-click and choose Arrange|by Category. The categories are listed alphabetically. You can collapse or expand the categories by clicking the **+** or **–** collapse icon and the state is persistent until you change it.

Note: Some properties occur in multiple categories. If you change the value under one category, the value changes consistently in all places.

Displaying properties alphabetically

To redisplay properties alphabetically, right-click and choose Arrange|by Name. The categories are no longer visible in the Object Inspector.

Jumping to the value of a component property

When the value of a property is another component, you can shift the Object Inspector's focus to that component by holding down the Ctrl key while double-clicking. For example, if the DataSet property of a data source is set to Table1, Ctrl-double-clicking on Table1 in the value column displays Table1's properties in the Object Inspector.

Property categories: functions and classes **(All editions)**

The following functions and classes have been added to support property categories in DesgnIntf.pas:

```
function RegisterPropertyInCategory(CategoryClass, PropertyName):
  TPropertyFilter;
function RegisterPropertyInCategory(CategoryClass, PropertyName,
  ComponentClass): TPropertyFilter;
function RegisterPropertyInCategory(CategoryClass, PropertyName,
  PropertyType): TPropertyFilter;
function RegisterPropertyInCategory(CategoryClass, PropertyType):
  TPropertyFilter;
```

This function has four formats. You can specify a category filter by property name; by class type and property name; by property type and property name; or by property type. Additionally, the property name can include wild card symbols. For example, you can add all properties that match 'Data*' to a particular category. For a full list of available wild card characters, see TMask.

```
function RegisterPropertiesInCategory(CategoryClass, array of consts):
  TPropertyCategory;
```

This function allows you to register a series of property names and/or property types filters in a single statement.

```
function IsPropertyInCategory(CategoryClass, ComponentClass, PropertyName):
  Boolean;
function IsPropertyInCategory(CategoryClass, ComponentClassName,
  PropertyName): Boolean;
```

This function has two formats. In either case, you can ask if a property of a certain class falls within the specified category. The class can be specified by name or class type.

```
function PropertyCategoryList: TPropertyCategoryList;
```

This function returns and creates, if necessary, the global property category list. See the next section.

Property categories: classes

The following components make up the category management system. While these components are publicly available, you should only access them through the above support functions.

`TPropertyCategoryList`

Contains and maintains the list of TPropertyCategories. There are numerous 'As a whole' access and manipulation methods for categories as well as simplified access functions.

`TPropertyCategory`

Contains and maintains the list of TPropertyFilters. There are numerous 'As a whole' access and manipulation methods for filters as well as data about the category itself.

`TPropertyFilter`

Maintains the information about a single filter associated with a particular category. Along with its filter-specific data, it also encapsulates the matching algorithm.

Images in drop down lists in the Object Inspector **(All editions)**

The Object Inspector now includes owner draw support to support customized rendering of properties. As a result, you can now see images in the drop down lists for properties that include images such as cursors, brush types, colors, and image lists. Many of these images appear in the Object Inspector by default without any change on your part. But note that to view images referenced by the ImageIndex property, you need to set the property that holds the image list to the image list containing the images.

Component writers can also use the owner draw support within components. For example, when writing property editors, you can now create owner draw lists of images to render images within the Object Inspector. See Owner draw support in the Object Inspector for details.

Owner draw support in the Object Inspector **(All editions)**

The following owner draw support is provided in a new property editor system:

- ListMeasureWidth
- ListMeasureHeight
- ListDrawValue
- PropDrawName
- PropDrawValue
- GetVisualValue

New overloadable methods on TPropertyEditor in DsgnIntf.pas

`Procedure TPropertyEditor.ListMeasureWidth(Value, Canvas, AWidth)`
This procedure is called during the width calculation phase of the drop down list preparation.

`Procedure TPropertyEditor.ListMeasureHeight(Value, Canvas, AHeight)`
This procedure is called during the item/value height calculation phase of the drop down list's render. This is similar to TListBox's OnMeasureItem, with slightly different parameters.

`Procedure TPropertyEditor.ListDrawValue(Value, Canvas, Rect, Selected)`
This procedure is called during the item/value render phase of the drop down list's render. This is similar to TListBox's OnDrawItem, but it has slightly different parameters.

`Procedure TPropertyEditor.PropDrawName(Canvas, Rect, Selected)`
This procedure is called during the render of the name column of the property list. Its functionality is similar to TListBox's OnDrawItem, but it has slightly different parameters.

`Procedure TPropertyEditor.PropDrawValue(Canvas, Rect, Selected)`
This procedure is called during the render of the value column of the property list. Its functionality is similar to PropDrawName. To determine what to render, this procedure should use the following function.

`Function TPropertyEditor.GetVisualValue: String;`
This function returns the displayable value of the property. If only one item is selected or all the multi-selected items have the same property value, this function returns the actual property value. Otherwise, this function returns an empty string.

New Project Manager features (All editions)

The Project Manager includes several new features to simplify managing the files particularly in large applications that contain many files and multiple projects within a project group:

- Project activation
- Project selector
- Drag and drop copying
- Copy and paste
- Remove button

You can also add additional types of files to your project (using drag and drop or Project|Add to Project) and view them in the editor as text files. You can also add resource files, and they are compiled into RES files and linked when you compile the project.

Project activation

Certain operations, such as commands available on context menus, operate on the active project. The active project is the one that is highlighted in bold in the Project Manager and is the project you are currently working on. The active project is also shown in the project selector.

When you view a project item, such as a form or a code file, the Project Manager automatically makes the project it belongs to the active project.

Project selector

You can easily select a project from the project group using the project selector at the top of the Project Manager. A drop-down list shows all projects in the current project group. The project you select becomes the active project.

Drag and drop copying

You can drag one or more selected items from any other Windows folder and drop them into a project in the Project Manager:

1. Activate the project where you want the copy to be placed.
2. In any Windows folder (such as the Windows Explorer or My Computer), select one or more items to copy.
3. Drag the item or items onto the name of a project in the Project Manager. (The project name is highlighted.)
4. Drop the items.

   You are asked to verify that you want to copy the item or items, and if you click Yes, the items are copied into the active project.
5. Save the project where you placed the copy.

Copy and paste

You can copy project items from one project to another:

1. Select the project item you want to copy.
2. Choose Edit|Copy (or type Ctrl+C).
3. Select the project where you want to place the copy (or move the cursor where you want to place the copy).
4. Paste the copy using Edit|Paste (or Ctrl+V).
5. Save the project where you placed the copy.

Remove button

The Remove button on the Project Manager toolbar removes the item which is selected. If a project is selected and you click on Remove, the whole project including all it contains is deleted from the current project group. If one file is selected, only that file is deleted when you click Remove. Realize that the files are not deleted from disk, only from the current project or group. The Project Manager verifies that you want to remove the project or item before doing so.

Note If you copy a project item into another then remove the first project without saving the second one, the copied item is removed from the second project as well. The Project Manager prompts you save the project before removing the item. If you save the project when prompted to do so, the copied item is retained.

TeamSource **(Enterprise or separate product)**

Note: TeamSource is a separate product and a separate installation, and is not provided with all versions of Delphi.

TeamSource is a new integrated workflow management tool for helping application development teams manage their daily tasks in a shared development environment. TeamSource uses a version control system for storing and retrieving shared files, but goes beyond simple version control to manage and coordinate the process of using a parallel model of source control.

For more information, see TeamSrc.hlp in the Help directory if TeamSource is available.

ActiveX enhancements **(Professional and Enterprise editions)**

The following enhancements were made to ActiveX:

- COM servers can be installed as components on the Component Palette for visual development. The Import Type Library option in the IDE allows you to install COM servers (such as Microsoft Word or Excel) as components. The component will expose the default events of the Server CoClasses as VCL events (accessible from the Events tab of the IDE's Property Inspector). Note that events with reference parameters are not hooked up for this release; the event will show up in the Property Inspector; however, the code to fire the Event is commented out.

  Numerous COM server components are provided on the new Servers tab of the Component Palette. You can use these components as if they were VCL components. For example, if you drag the Microsoft word document components onto a form, the following code brings up an instance of Microsoft Word.

```
WordApplication1.Connect;
Wordapplication1.Visible:=True;
```

- HResult was changed from an unsigned int to a signed int.
- An ActiveServerPage wizard was developed.You can now create Automation Objects that, when invoked from an .ASP page on an IIS server, have access to the various interfaces representing a user request, response, etc. A sample .ASP illustrating the syntax to create the server can also be generated for you.
- An Apply Updates dialog was added for COM servers. The dialog displays the changes made to your CoClass implementation file by the IDE as you modify your CoClass interfaces in the Type Library Editor. You may veto some or all of the proposed changes.
- Free-threading support was added for COM factories.
- The COM Object wizard has a new checkbox that allows you to mark the object's default interface as OLEAUTOMATION. This flag allows your interface to be marshaled by the Type Library Marshaler, obviating the need for a proxy-stub DLL for custom interfaces.

Note: You'll need to ensure that your interface utilizes OLE Automation compatible types.

- Delphi supports sparse vtables created in Visual Basic. When importing an Active Server or Active Control created in Visual Basic, Delphi will detect any gaps in the interfaces of the server and automatically insert dummy entries in the interface definition generated in the xxxx_TLB file.

Editor enhancements **(All editions)**

The following enhancements were made to the editor:

- Customizable editor properties are now centralized under Tools|Editor Options.
- A new editor keymapping was added for Visual Studio emulation.
- A new Key Mappings tab was added where you can enable customized key mapping modules and enhancement modules.

Using the Open Tools API enhancements, you can customize key bindings by adding a completely new key mapping module plus all the hot keys in the IDE or you can create an enhancement module to replace a few keys.

Examples for both full and partial key bindings are provided in Demos\ToolsAPI\Editor Keybinding. The full example is called New IDE Classic where new keys are bound to existing functionality. The partial example is called Buffer List, which adds a utility that displays the buffer list, to a hot key (Ctrl+B). You can load them using the Key Mappings tab of the Tools|Editor Options.

International tools **(Enterprise edition)**

The Integrated Translation Environment (ITE), available in some versions of Delphi, is a suite of tools for software localization and simultaneous development for different locales. It is integrated with the IDE to let you manage multiple localized versions of an application as part of a single project. The ITE includes three tools:

ITE tool	Description
Translation Manager	Displays a grid for viewing and editing translated resources
Translation Repository	Provides a database for translations that can be shared across projects and by different developers
Resource DLL wizard	An improved DLL wizard that generates and manage resource DLLs

The ITE provides an Active Language setting to let you run localized versions of an application in debug mode.

You can now compile RC files from the Delphi IDE. There is no need to invoke the Resource Compiler from the command line.

Refer to the ITE overview for details about this localization technology.

Help menu changes **(All editions)**

The Delphi Help menu now divides the Help system into three major areas:

Help command	Contents
Delphi Help	Opens DELPHI5.HLP and accesses all the main Delphi Help files including help for the VCL and the IDE.
Delphi Tools	Opens DEL5XTRA.HLP and accesses detailed help for tools such as SQL Builder, SQL Monitor, Database Explorer, Winsight, Image Editor, and TeamSource.
Windows API/SDK Help	Opens WIN32SDK.HLP and accesses Microsoft's Help for objects that you can use in Delphi. Note that Installation of the Microsoft Help files is optional. This option only appears if they are installed.

Upgrading to Delphi 5

When you load a Delphi 4 project into Delphi 5, it is automatically updated. The following topics describe changes that could potentially impact existing Delphi projects:

- Summary of compatibility issues
- Converting MIDAS applications
- Changes to MIDAS support

Refer also to the What's new for information on additional features that you may want to incorporate into your applications.

Summary of compatibility issues

Following are general compatibility issues that may affect your Delphi applications:

- Forms are now saved as text. See Forms in Delphi 5.
- Any property that represents an index into an image list has changed from type Integer to type TImageIndex.
- The type of the TDatabase OnLogin event has been renamed TDatabaseLoginEvent from TLoginEvent.
- COM changes to CoInitFlags have occurred to support initialization and multi-threading. See CoInitFlags changes.
- The HRESULT type is declared in System.pas as a signed 32-bit integer (Longint), which matches the Windows SDK. In Delphi 4, but not in previous versions, HRESULT had been declared in Ole2.pas as an unsigned 32-bit integer (Longword).
- If you are responsible for upgrading multi-tier client/server applications, MIDAS has changed. See Converting MIDAS applications, Changes to MIDAS support, and Changes to MIDAS security for information on how to update your MIDAS applications from previous releases.
- The default alignment has changed for this release. See Default alignment for an example of how this might affect your application.
- TCustomTreeview.CustomDrawItem has a new parameter. If an existing application overrides this method, it will not compile until a new parameter is added.
- The TWebBrowser component on the Internet page replaces the THTML component from Netmasters. See Upgrading applications that use THTML.
- Delphi 4 symbol files are not compatible with Delphi 5. If you see the message “Error reading symbol file” when opening a Delphi 4 application, close the message box and rebuild the application.
- When you create an application, Delphi creates a <project>.RES file which contains version info resources plus the application’s main icon. As of Delphi 5, .RES files are required to open a project. If a .RES file is not found, you will receive an error message (File not found <project>.RES). If you are opening an older project that does not have an associated .RES file, you can copy a .RES file from another project (such as those supplied in your \Help\Examples folders) to your project location, change the base file name of the "dummy" .RES to match your project name, then reopen your project. Your project will update the acquired .RES file as necessary.
- Existing controls that invoke pop-up menus in response to WM_RBUTTONUP or OnMouseUp events may exhibit "double" pop-up menus or no pop-up menus at all when compiled with Delphi 5. See Changes with invoking pop-up menus.
- You can use packages when building an ISAPI/NSAPI DLL. This change impacts existing WebBroker applications. See Upgrading WebBroker applications.
- DLL startup code no longer sets the FPU control word. See Changes to DLL initialization code.
- When you export an overloaded function or procedure from a DLL, you must specify the routine's parameter list in the **exports** clause. As a result, some code written for early releases of Delphi 4 will cause compilation errors in Delphi 5. For more information, see The exports clause.
- Code generation when importing type libraries has been modified and enhanced to allow you to map symbol names. See Mapping symbol names in the type library.
- The global FMTBCDToCurr and CurrToFMTBCD routines have been replaced by the new BCDToCurr and CurrToBCD routines (and the corresponding protected methods on TDataSet have been replaced by the protected and undocumented DataConvert method).
- The TFieldType enumerated variable has additional values.
- For QuickReports printing, the format of the method BeforePrint changed from

```
BeforePrint(Sender: TQuickRep; var PrintReport: Boolean);
```

  to

```
BeforePrint(Sender: TCustomQuickRep; var PrintReport: Boolean);
```

- The TwoDigitYearCenturyWindow variable now has the default value of 50 (instead of 0). This affects the way 2-digit string dates are converted to numeric dates with StrToDate or StrToDateTime. For example, 00 converts to 2000 instead of 1900, but 50 or later will still be 1950.
- The new Resource DLL wizard does not fully process output from the old Resource DLL wizard. If you run the wizard to update an old translation project, existing translations from form files are placed in

the Translation Manager's "Previous ..." column. Existing translations from other resource files are not preserved, but you can copy them from the old PRJSTRS.RC file.

- Under Web Deployment options, Delphi no longer supports code signing. Developers who want to code sign their files must now use the command-line utilities provided by Microsoft.

Forms in Delphi 5

Form files (DFMs) are now saved as text rather than binary, by default. Therefore, if you plan to use forms created in Delphi 5 in earlier versions of Delphi, you need to save the forms as binary. Right-click on a form and uncheck Text DFM to save a particular form as binary. Uncheck the New forms as text on the Preferences page of the Tools|Environment Options to save forms in binary as your default modus operandi.

In addition, if you plan to use Delphi 5 forms in earlier versions of Delphi, using new or changed properties may cause errors when opening the forms particularly if you set properties to other than the default value. (For example, if using the new published properties THeaderSection.ImageIndex and THeaderControl.DragReorder. In short, be careful not to use new features in your forms if you need them to be backward compatible.

CoInitFlags changes

The class factories in Delphi 5 better support COM initialization and multi-threaded memory management. In Delphi 4, the only way to make a free threaded EXE COM server was to initialize CoInitFlags in your program's main source file (.dpr in Delphi) before the call to Application.Initialize. Delphi 5 makes that step unnecessary: if your exe contains a free threaded server, Delphi will initialize COM for freethreading.

This change should not adversely affect apartment-threaded objects in EXE servers that contain a mix of free threaded and apartment threaded objects. Even though COM is initialized at the project level for free threading, your apartment threaded objects will still be treated as apartment model by COM, and if you create new threads for apartment model objects, you're responsible for calling CoInitialize anyway.

If you have an application with multiple COM objects, registration occurs using the highest threading model support. For example, if you have one tmFree object and one tmSingle object, the CoInitFlags is set to COINIT_MULTITHREADED and IsMultiThread is set to True.

CoInitFlags only affects COM EXE servers. Therefore, if in Delphi 4 you developed a COM EXE server with an object marked with the tmFree ThreadingModel and didn't change the CoInitFlags:

- In Delphi 4, you would **not** have a free threaded server
- In Delphi 5, you **would** have a free threaded server

The only problem this might cause is for objects marked for free threading that never actually executed as free threaded in Delphi 4 if you neglected to initialize CoInitFlags yourself. In Delphi 5, those objects will now execute as free threaded. If the objects are not fully prepared for free threading resource protection, the objects could exhibit "new" errors in code that previously appeared to work (because it wasn't actually free threaded). If you are uncertain about how this change affects your objects, change the threading model of your objects to Apartment to preserve their Delphi 4 behavior.

CoInitFlags

If you have a COM object with a ThreadingModel of tmFree or tmBoth, CoInitFlags are set to COINIT_MULTITHREADED. For a COM object with a ThreadingModel of tmApartment, CoInitFlags are set to COINIT_APARTMENTTHREADED.

Use of the ThreadedClassFactory is no longer needed. It may interfere with COM reference counting.

IsMultiThreaded

If you have a COM object with a ThreadingModel of tmApartment, tmFree, or tmBoth, IsMultiThreaded is set to True.

Summary

ThreadingModel	IsMultiThread	CoInitFlags
tmFree	Yes	COINIT_MULTITHREADED
tmBoth	Yes	COINIT_MULTITHREADED
tmApartment	Yes	COINIT_APARTMENTTHREADED
tmSingle	No	CoInitialize(nil) is called (like COINIT_APARTMENTTHREADED)

Default alignment

The compiler's default data alignment has changed from packed to unpacked for Microsoft compatibility. Because the compiler's default data alignment has changed, this section provides details on data alignment and potential impacts on your Delphi applications.

What is data alignment?

Setting data alignment affects the size and placement of the data members of structures and classes. Thus, if a pointer to a structure is passed to a DLL or another OBJ that has been compiled with a different data alignment, the destination code could be referencing incorrect locations in the struct or class for data. This can lead to strange behavior when the program is executed or potentially cause it to crash.

The problems caused by having mismatched data alignment are commonly hard to track down. For example:

```
var
   T: record
     ld1: extended;
     ld2: extended;
   end;
```

Compiled as unaligned or packed (using $a-):T.ld1 resides in bytes 0 through 9 of T, T.ld2 resides in bytes 10 through 19. No padding is necessary for packed alignment. The record is 20 bytes in size.

Compiled aligned or unpacked (using $a+): T.ld1 resides in bytes 0 through 9 of T, T.ld2 resides in bytes 16 through 25. Because the structure needs to be padded with empty bytes for unpacked alignment, the record is 32 bytes in size.

Converting MIDAS applications

The architecture to support multi-tier database (MIDAS) servers in stateless environments (such as MTS) has changed in Delphi 5. This has advantages over the previous architecture and offers a significant increase in performance and scalability by reducing message traffic. You will need to update your MIDAS 1 and MIDAS 2 applications to work with MIDAS 3.

Follow these steps to convert your MIDAS 1 or MIDAS 2 applications to MIDAS 3:

1. Open your MIDAS server.
2. View the Type Library.
3. Add the Borland Midas type library to the Uses page of the server library properties.
4. Remove the BdeProv unit from the uses clause.
5. Change the Parent Interface of your server to IAppServer.
6. Write down the name of any property that returns an IProvider and note any other methods that use IProvider. IProvider is no longer supported, so delete all properties that return an IProvider.

   Note: You need to rewrite methods that use IProvider. These properties are added when you export a provider and may include custom methods that use IProvider as well.
7. Save the type library.
8. Open your RemoteDataModule.
9. From the list that you created of old IProvider properties, make sure you have a TProvider for each of those properties. If any are missing, drop a TProvider and set its Name property to the old property name and connect it to the appropriate dataset.

   Note: If porting a MIDAS 1 application, you may want to take this opportunity to convert your master detail relations into nested datasets.
10. For any TProvider that you don't want visible from the client, set TProvider's Exported property to False.
11. Delete the same IProvider properties from the data module that you deleted from the type library.
12. If your data module is derived from TDataModule, change it to derive from TRemoteDataModule.
13. Run the Socket Server and choose Connections|Registered Objects Only to disable added socket and web connection security. See Changes to MIDAS security for information on how to update your application to include this security.
14. Recompile the server.
15. Recompile the client.

Refer also to Changes to MIDAS support for detailed information about the architectural changes.

Changes to MIDAS support

The following changes were made to support MIDAS enhancements. See Converting MIDAS applications for information on how to how to update your MIDAS applications.

TDataSet

Provider property was removed. You now need to use an explicit provider component for any dataset.

Socket and web connection registration

The Socket Server used to make all objects available. Now only those objects that have been properly registered will be made available in the socket connection or web connection dropdown boxes. See Changes to MIDAS security for information on how to update your application to override this security or to determine how to register application servers.

IProvider

IProvider was removed. You can now use the AppServer property on TClientDataSet or the GetServer method of the TCustomRemoteServer (Connection) components. Here is the list of replacements showing the old IProvider functions with a code sample, then the new function with a new code sample.

Constraints

Constraints cannot be turned on or off from the client. These methods no longer work.

```
function IProvider.Get_Constraints: WordBool; safecall;
procedure IProvider.Set_Constraints(Value: WordBool); safecall;
property IProvider.Constraints: WordBool read Get_Constraints write
Set_Constraints;
```

Getting and setting properties

The process of getting data and setting properties such as Params and Metadata have changed. Typically, you won't have to do anything different. If you used to call any of these methods directly from the IProvider, then you will now need to call them on IAppServer.

```
function IProvider.Get_Data: OleVariant; safecall;
function IProvider.GetMetaData: OleVariant; safecall;
function IProvider.GetRecords(Count: Integer; out RecsOut: Integer):
OleVariant; safecall;
procedure IProvider.SetParams(Values: OleVariant); safecall;
procedure IProvider.Reset(MetaData: WordBool); safecall;
property IProvider.Data: OleVariant read Get_Data;


function GetDataTheOldWay(Connection: TDCOMConnection; ProviderName:
string; Params: OleVariant;
    Metadata: Boolean);
var
  Provider: IProvider;
  RecsOut: Integer;
begin
  Provider := Connection.GetProvider(ProviderName);
  Provider.Reset(Metadata);
  Provider.SetParams(Params);
  Result := Provider.GetRecords(-1, RecsOut);
  { 5 network roundtrips, 1 additional interface }
end;


function AS_GetRecords(const ProviderName: WideString; Count: Integer;
out RecsOut: Integer;
  Options: Integer; var Params: OleVariant; var OwnerData: OleVariant):
```

```
OleVariant; safecall;


function GetDataTheNewWay(Connection: TDCOMConnection; ProviderName:
string; Params: OleVariant;
  Metadata: Boolean);
var
  Options, RecsOut: Integer;
begin
  if Metadata then Options := INCLUDE_METADATA else Options := 0;
  Result := Connection.GetServer.AS_GetRecords(ProviderName, -1,
RecsOut, Options, Params, NULL);
  { 1 network roundtrip, 0 additional interfaces }
end;
```

ApplyUpdates

You can still use the ApplyUpdates from the TClientDataSet. If you were calling ApplyUpdates directly from the IProvider, you need to use the IAppServer instead.

```
function IProvider.ApplyUpdates(Delta: OleVariant; MaxErrors: Integer;
out ErrorCount: Integer): OleVariant; safecall;


procedure ApplyUpdatesTheOldWay(CDS: TClientDataSet);
var
  ErrorPacket: OleVariant;
  Errors: Integer;
begin
  ErrorPacket := CDS.Provider.ApplyUpdates(CDS.Delta, -1, Errors);
  if Errors > 0 then
    CDS.Reconcile(ErrorPacket);
end;


function AS_ApplyUpdates(const ProviderName: WideString; Delta:
OleVariant; MaxErrors: Integer;
  out ErrorCount: Integer; var OwnerData: OleVariant): OleVariant;
safecall;


procedure ApplyUpdatesTheNewWay(CDS: TClientDataSet);
var
  ErrorPacket: OleVariant;
  Errors: Integer;
begin
  ErrorPacket := CDS.AppServer.AS_ApplyUpdates(CDS.ProviderName,
CDS.Delta, -1, Errors, NULL);
  if Errors > 0 then
    CDS.Reconcile(ErrorPacket);
end;
```

DataRequest

You can now use a DataRequest helper function on TClientDataSet or call it directly from the IAppServer interface.

```
function IProvider.DataRequest(Input: OleVariant): OleVariant; safecall;


procedure DataRequestTheOldWay(CDS: TClientDataset);
begin
  CDS.Provider.DataRequest('MyCustomData');
```

```
end;

function IAppServer.AS_DataRequest(const ProviderName: WideString; Data:
OleVariant): OleVariant;

procedure DataRequestTheNewWay(CDS: TClientDataset);
begin
  CDS.DataRequest('MyCustomData');
{ or }
  CDS.AppServer.AS_DataRequest(CDS.ProviderName, 'MyCustomData');
end;
```

TClientDataSet

FetchParams still works as it did, except, you no longer have to call it after executing a stored procedure with output parameters. The output parameters are automatically updated on the TClientDataSet.

Many BeforeXXX and AfterXXX events have been added to allow custom data to be sent with calls to the server. This allows stateless servers to maintain state information on the client.

HasProvider is now HasAppServer.

The Provider property is now the AppServer property.

An Execute method has been added to allow StoredProcedure and Queries that do not return a result set to be executed.

SendParams has been removed.

TDataSetProvider

TDataSetProvider can now apply updates directly to the database server as well as to a dataset. The dataset components implement an interface that enables them to work with a dataset provider. This allows TDataSetProvider to work with any sort of dataset (BDE-enabled, InterBase Express, or ADO-based as well as client datasets or custom datasets) while not requiring the BDE or DBOLE (the requirement for a specific data access engine rests with the dataset, not the provider).

Because of these changes, the distinction between TProvider and TDataSetProvider is no longer relevant. TDataSetProvider can do everything previously handled by TProvider and more. As a result, TProvider has been removed from the component palette. The class declaration still exists (although it has moved from Bdeprov to Provider), but only for backward compatibility.

TCustomRemoteServer or TXXXConnection components

GetProvider has been removed.

GetServer now returns an IAppServer interface that can be used to call methods on the server.

TCustomProvider

Many BeforeXXX and AfterXXX events have been added to allow custom data to be sent with calls to the server. This allows stateless servers to maintain state information on the client.

TCustomProvider.Reset no longer exists. It now is an option in GetRecords. (The interface of GetRecords has changed.) TCustomProvider.FetchData also no longer exists.

Changes to MIDAS security

Security has been added to the socket and web connection. The server will only show objects that have been exported for that connection type. You may want to upgrade your application server to include a call to enable socket transport. You can also use the Socket Server menu item (Connections|Registered Objects Only) to disable security checking (for backward compatibility). This option is ON by default.

If upgrading your application: objects are exported by setting registry entries. Under CLSID\{xxx}, values for Sockets and Web must be set to 1 for exported. You can also use the following registry helper functions to register these values:

- EnableSocketTransport
- DisableSocketTransport
- EnableWebTransport
- DisableWebTransport

By default, TRemoteDataModule includes an overriden UpdateRegistry method that looks like this:

```
class procedure TMyRDM.UpdateRegistry(Register: Boolean; const ClassID,
ProgID: string);
begin
  if Register then
  begin
    inherited UpdateRegistry(Register, ClassID, ProgID);
    EnableSocketTransport(ClassID);
    EnableWebTransport(ClassID);
  end else
  begin
    DisableSocketTransport(ClassID);
    DisableWebTransport(ClassID);
    inherited UpdateRegistry(Register, ClassID, ProgID);
  end;
end;
```

This means that by default both transports are enabled. You can remove these lines from this method to disable an object for a particular transport.

Upgrading applications that use THTML

To upgrade applications from prior releases of Delphi and which use THTML, you have two choices:

- Use the new TWebBrowser component on the Internet page of the Component Palette.
- Import the Delphi 4 NetMasters HTML.OCX (located on the Delphi 5 CD-ROM in \Info\Extras\NetManage).

To install NetManage OCX components:

1. Copy the files Html.ocx, *.dll to the Windows\System directory.
2. Register the Html.Ocx control.
3. From within Delphi, install the package \Bin\DCLIsp50.bpl.

Mapping symbol names in the type library

When importing type libraries in Delphi 5, custom mapping for some type libraries is specified in tlibimp.sym and may differ from code generation in previous releases. For example, when doing Microsoft Word automation, the symbol which used to be CoApplication_ is now CoWordApplication.

In previous releases, when new type libraries were added using an existing identifier, an underscore was added to distinguish the identifier. So, the identifier called Application became Application_. Multiple underscores might be added if another imported type library used the same identifier.

Delphi 5 provides the ability to map symbol names in the type library within the tlibimp.sym file located in the BIN directory. The tlibimp.sym file contains mapping for some common applications including Microsoft Word, Microsoft Excel, Microsoft PowerPoint, and Microsoft Access. If you have applications that use these servers, you may have conflicts when compiling them in Delphi 5.

The following example shows how symbol names are remapped in Microsoft Word. The first line (after the comment) specifies the GUID (globally unique identifier) of the server in the registry. For example,

```
[{00020905-0000-0000-C000-000000000046}:TypeNames]
```

The rest of the lines remap keys to new values (instead of just adding an underscore to conflicting identifiers). For example,

```
Application=WordApplication
```

Example remapping for Microsoft Word

```
;;=============================================;;
;; Map WinWord CoClasses to better names        ;;
;;=============================================;;
[{00020905-0000-0000-C000-000000000046}:TypeNames]
Application=WordApplication
Document=WordDocument
Font=WordFont
ParagraphFormat=WordParagraphFormat
OLEControl=WordOLEControl
LetterContent=WordLetterContent
```

You can map conflicting symbols for other servers whose type libraries you want to import by using the same format as shown for the example above.

Note Be careful when editing tlibimp.sym. Don’t change remapping that occurs for C++ or Pascal member or type names in the beginning of the file.

Changes with invoking pop-up menus

Delphi 5.0 now relies on the WM_CONTEXTMENU message to invoke pop-up menus. Windows sends controls WM_CONTEXTMENU in response to WM_RBUTTONUP messages, Shift+F10 keystroke messages, and "Windows Key" keystroke messages on special keyboards. Accessibility options can enable other keystrokes or devices to invoke context sensitive help on a control.

Existing controls that invoke pop-up menus in response to WM_RBUTTONUP or OnMouseUp events may exhibit "double" pop-up menus or no pop-up menus at all when compiled with Delphi 5. Controls should now invoke pop-up menus in response to WM_CONTEXTMENU messages, or in the new OnContextPopup event.

Existing controls that process WM_RBUTTONUP messages MUST pass the message to the inherited message handler (or DefWndProc) to enable the default handlers to issue a WM_CONTEXTMENU message.

The move to WM_CONTEXTMENU may inconvenience some existing controls, but we feel this change is necessary to support the increasing number of ways that the Win32 UI provides to invoke context menus by mouse, keyboard, and accessibility devices.

Upgrading WebBroker applications

WebBroker now supports the use of packages when building an ISAPI/NSAPI DLL. To do this, several items were moved from the HTTPApp unit into a new unit called WebBroker.

The existing units ISAPIApp and CGIApp, along with the new WebBroker unit, include the compiler directive {$DENYPACKAGEUNIT}. This ensures that these units will never be allowed to be contained within a package. These units will always be linked directly into the WebBroker application (EXE, DLL).

If you want to use the WebBroker architecture and rely on the WebBroker's Application variable, you must include WebBroker in the uses clause. Otherwise, you will get an undeclared identifier error.

To upgrade existing Delphi WebBroker applications (ISAPI/NSAPI, CGI, and WinCGI):

- Replace the reference to HTTPApp in the uses clause in the main program .DPR file with WebBroker.

Changes to DLL initialization code

Changes in DLL initialization code make Delphi DLLs more compatible with host applications that do not support floating-point exceptions or full floating-point precision. The DLL initialization code now preserves the FPU control word established by the host application.

In previous versions of Delphi, DLLs set the FPU control word to enable floating-point exceptions for operations (such as division by zero) and to enable full 80-bit precision in calculations. Some DLL hosts, including scripting languages in web browsers, can crash or fail when floating-point exceptions are turned on when they load a DLL or instantiate an in-process COM server.

In Delphi 5, DLLs no longer assert the Delphi-preferred FPU control word ($1332). This means that when your code resides in a DLL, the precision of FPU calculations and exception handling behavior will depend on the host application's preferences. This may adversely affect your existing Delphi code that relies on full precision calculations or FPU exceptions.

If you require a particular FPU configuration in DLL routines, you should set the FPU control word yourself (call Set8087CW), preferrably immediately before the code that needs the particular FPU configuration. To avoid disrupting the host application, you should restore the FPU control word to its original value as your routine returns to its caller.

Link not found

The topic you requested is either not available or not linked to this Help system. This can occur if you launched this Help file from a system on which Delphi has not yet been installed, or if the subject matter you are requesting is not available in your edition of Delphi.

Borland INPRISE

The topic you requested is now loading. If it does not appear within a few seconds, the topic is either not available or not linked to this Help system. This can occur if you launched this Help file from a system on which Delphi has not yet been installed, or if the subject matter you are requesting is not available in your edition of Delphi.