Report Builder
User’s Guide
# Table of Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foreword</td>
<td>0</td>
</tr>
<tr>
<td><strong>Part I Report Builder</strong></td>
<td>5</td>
</tr>
<tr>
<td>1 Overview</td>
<td>5</td>
</tr>
<tr>
<td>2 How To</td>
<td>8</td>
</tr>
<tr>
<td>Template Files</td>
<td>8</td>
</tr>
<tr>
<td>Add Fields</td>
<td>9</td>
</tr>
<tr>
<td>Receive Filters</td>
<td>11</td>
</tr>
<tr>
<td>Adjust Search</td>
<td>12</td>
</tr>
<tr>
<td>More Help</td>
<td>13</td>
</tr>
<tr>
<td>3 Workspaces</td>
<td>14</td>
</tr>
<tr>
<td>About the Report Designer</td>
<td>14</td>
</tr>
<tr>
<td>Data Workspace</td>
<td>14</td>
</tr>
<tr>
<td>Design Workspace</td>
<td>15</td>
</tr>
<tr>
<td>Preview Workspace</td>
<td>16</td>
</tr>
<tr>
<td>4 Data-Specific Tools</td>
<td>17</td>
</tr>
<tr>
<td>Database</td>
<td>17</td>
</tr>
<tr>
<td>DataView</td>
<td>18</td>
</tr>
<tr>
<td>Data pipeline</td>
<td>19</td>
</tr>
<tr>
<td>SQL</td>
<td>19</td>
</tr>
<tr>
<td>Query Wizard</td>
<td>19</td>
</tr>
<tr>
<td>Query Designer</td>
<td>23</td>
</tr>
<tr>
<td>5 Report-Building Tools</td>
<td>27</td>
</tr>
<tr>
<td>About report-building tools</td>
<td>27</td>
</tr>
<tr>
<td>Report Wizard</td>
<td>28</td>
</tr>
<tr>
<td>Label Template Wizard</td>
<td>34</td>
</tr>
<tr>
<td>CrossTab Wizard</td>
<td>35</td>
</tr>
<tr>
<td>6 Toolbars</td>
<td>36</td>
</tr>
<tr>
<td>About toolbars</td>
<td>36</td>
</tr>
<tr>
<td>Advanced Component Palette</td>
<td>36</td>
</tr>
<tr>
<td>Align or Space Toolbar</td>
<td>37</td>
</tr>
<tr>
<td>Data Component Palette</td>
<td>37</td>
</tr>
<tr>
<td>Data Tree</td>
<td>37</td>
</tr>
<tr>
<td>Draw Toolbar</td>
<td>38</td>
</tr>
<tr>
<td>Standard Toolbar</td>
<td>38</td>
</tr>
<tr>
<td>Edit Toolbar</td>
<td>38</td>
</tr>
<tr>
<td>Format Toolbar</td>
<td>40</td>
</tr>
<tr>
<td>Nudge Toolbar</td>
<td>40</td>
</tr>
<tr>
<td>Report Tree</td>
<td>40</td>
</tr>
<tr>
<td>Standard Component Palette</td>
<td>41</td>
</tr>
<tr>
<td>7 Bands</td>
<td>41</td>
</tr>
<tr>
<td>About Bands</td>
<td>41</td>
</tr>
<tr>
<td>Header</td>
<td>42</td>
</tr>
<tr>
<td>Group</td>
<td>43</td>
</tr>
<tr>
<td>Group Header</td>
<td>43</td>
</tr>
<tr>
<td>Detail</td>
<td>44</td>
</tr>
<tr>
<td>Footer</td>
<td>45</td>
</tr>
</tbody>
</table>

© 2004 - 2007 Express Technology Inc.
8 Components

Advanced
- CrossTab
- Region
- SubReport

Standard
- BarCode
- Checkbox
- Image
- Label
- Line
- Memo
- RichText
- System Variable
- Shape
- Variable

Data-aware
- DBBarCode
- DBCalc
- DBCheckBox
- DBImage
- DBMemo
- DBRichText
- DBText

Speed Menu Options
- AutoSize
- Bring to Front
- Configure
- DisplayFormat
- Edit
- Lines
- MailMerge
- MaintainAspectRatio
- Pagination
- ParentHeight
- ParentWidth
- Position
- ReprintOnOverflow
- Send to Back
- ShiftRelativeTo
- StretchWithParent
- StretchWithParent
- Stretch
- Style
- SuppressRepeatedValues
- Visible

8 Components

Static VS. Stretchable
- Stretchable
- Static

9 Reference
# Glossary

- Standard Routines .................................................................................................................. 83
- Debugging Options .................................................................................................................. 79
- The Code Toolbox ..................................................................................................................... 76
- Events .......................................................................................................................................... 73
- Declaring Global Variables ...................................................................................................... 75
- Programs in RAP ....................................................................................................................... 74
- The Code Toolbox ..................................................................................................................... 76
- Data Tab ...................................................................................................................................... 77
- Objects Tab ................................................................................................................................. 77
- Language Tab ............................................................................................................................. 78
- Debugging Options ..................................................................................................................... 79
- CodeSite Support ....................................................................................................................... 79
- CodeSite Functions ..................................................................................................................... 80
- Using the CodeSite Functions .................................................................................................... 81
- Conditionally Compiling CodeSite Support ............................................................................... 82

# 10 RAP Reference

## What is RAP

- Overview of Features ............................................................................................................... 61
- Overview of the Interface ......................................................................................................... 62
- The Calc Tab ............................................................................................................................. 62
- The Code Explorer .................................................................................................................... 63
- The Code Editor ....................................................................................................................... 64
- The Code Toolbox ..................................................................................................................... 65
- The Message Window .............................................................................................................. 66
- The Variables View ................................................................................................................... 66
- The Events View ....................................................................................................................... 66
- The Module View ..................................................................................................................... 67
- Context-Sensitive Help ............................................................................................................. 69

## Quick Start Tutorials

- Color-coding a DBText Component ....................................................................................... 71
- Concatenating Fields ............................................................................................................... 71
- Dynamic Duplexing .................................................................................................................. 71
- Adding New Functions to RAP ............................................................................................... 72
- Extending the RAP RTTI .......................................................................................................... 72
- Printing a Description of AutoSearch Criteria ..................................................................... 72
- Displaying Delphi Forms From RAP .................................................................................... 72

## Programming with RAP

- Procedures and Functions ...................................................................................................... 73
- Declaring Local Variables ...................................................................................................... 73
- Declaring Local Constants ..................................................................................................... 73
- Calling Procedures and Functions ......................................................................................... 73
- Procedure and Function Parameters ...................................................................................... 73
- Events .......................................................................................................................................... 73
- Coding an Event Handler ........................................................................................................ 73
- Compiling Event Handlers ...................................................................................................... 74
- Globals: The Module View ..................................................................................................... 74
- Programs in RAP ....................................................................................................................... 74
- Declaring Global Variables .................................................................................................... 75
- Declaring Global Constants ................................................................................................... 75
- Declaring Global Procedures and Functions ....................................................................... 75
- The Code Toolbox ..................................................................................................................... 76
- Overview of the Code Toolbox .............................................................................................. 76
- Data Tab ...................................................................................................................................... 77
- Objects Tab ............................................................................................................................... 77
- Language Tab ........................................................................................................................... 78
- Debugging Options .................................................................................................................. 79
- CodeSite Support .................................................................................................................... 79
- CodeSite Functions .................................................................................................................. 80
- Using the CodeSite Functions ............................................................................................... 81
- Conditionally Compiling CodeSite Support ....................................................................... 82

## Language Reference

- RAP Language Overview ........................................................................................................ 83
- Standard Routines .................................................................................................................... 83
- String Functions ....................................................................................................................... 83
- Conversion Functions ............................................................................................................... 84
- Format Functions ..................................................................................................................... 84
- DateTime Functions .................................................................................................................. 84

© 2004 - 2007 Express Technology Inc.
1 Report Builder

INTRODUCTION- REPORT BUILDERS

ExpressMaintenance includes an extremely powerful built-in Report Builder. Report Builder unleashes your data to be reported and exported virtually any way you can imagine.

Report Builder is developed by Digital Metaphors Corporation and offers ExpressMaintenance users the maximum in data reporting flexibility. This help section was designed by Digital Metaphors Corporation and is made available through the ExpressMaintenance help system.

Designing custom reports does require some familiarity with database concepts. Using the report builder is an easy process but may take a little practice. We recommend you start with simple list reports and then work up to more complicated reports.

You will find this help to covers the Report Builder features and tools. However, it does not address the concepts of data structures. Most of the data will be self explanatory since it is your maintenance data. Additional help is available.

For additional help:
- Do not contact Digital Metaphors directly as they do not provide end user tech support.
- Visit the Report Builder site to learn more about Report Builder at: http://digital-metaphors.com/
- Consult this help file completely and practice with simple reports.
- Download, install the Report Builder Learning System provided by Digital Metaphors and available from our website at: http://www.expresstechnology.com/DownloadTrial.htm
- Contact Express Technology tech support via email for specific help at: support@ExpressTechnology.com

1.1 Overview

OVERVIEW

In an overview to ReportBuilder, the reporting equation is described. The reporting equation divides reporting into four main activities:

With ReportBuilder Pro, you get a complete set of components that enable end-users to perform the data access, report layout, and report output activities of the reporting equation. This goal is achieved by delivering visual, easy-to-use solutions. This screen shot of the ReportBuilder Pro Report Designer shows the ergonomic design of the user-interface.

Data

Within the work environment of the Data tab, end users can quickly create dataviews, which can then be used to supply data to reports. Dataviews are usually created via the Query Wizard or Query Designer. Both of these tools are visual; they also allow the end-user to select the tables, fields, search criteria, and sort order necessary for the report. Behind the scenes, an SQL statement is generated and used to retrieve the data from the database. A screen shot of a
completed dataview is shown below.

The solution described above is the standard behavior within the data workspace. However, the developer can customize this user-interface by doing one of three tasks:

- Register a replacement query wizard or query designer.
- Remove the query wizard or query designer.
- Create new dataview template classes that can simplify the data selection process even further by establishing the relationship between the tables in the database and presenting an alternative user-interface (such as a single form that allows search/sort criteria to be entered).

The bottom-line is that the Data area contains a turnkey solution that can be used out-of-the-box, but if customizations are needed, an architecture has been provided so that those customizations are possible.

**Design**

The Design workspace contains the actual layout of the report. The user-interface is identical to the one presented to developers using ReportBuilder at Delphi design-time; in other words, it is full-featured and professional. The Office97 interface-style makes the Design workspace especially easy to learn for end users. A Report Wizard is available for creating reports quickly. You can customize this interface by replacing any of the dialogs it uses and by registering your own report wizards.
Preview
The Preview workspace contains the rendered report. The report can be printed to the printer or to various file formats from this workspace.
1.2 How To

HOW TO ...

Below are common questions presented by users concerning the use of the Report Builder:

- How do I import / export report templates into the application?
- How do I add fields or adjust the layout of a report?
- How can I make my report receive filters from ExpressMaintenance?
- How to make a report include all of the data records I expect?
- How can I get comprehensive help or a tutorial on Report Builder?

You can download original report definitions for all reports - click here for list.

1.2.1 Template Files

How do I import / export report templates into the application?

All Express Technology applications have a feature that allows users to import and export report formats to report template files. Exporting a report to file is very helpful if you want to save a report
template to a text (.rtm, .rdf*, .ini or .txt) file before making alterations. Or, you might want to export a report template to a text file and email it to our support staff.

Occasionally, our support staff may make a custom alteration to a report and email it to you for importing. Or, you may want to import one of the original report formats.

The text file produced by exporting reports contains the report template / layout. It does not contain data but rather the report setup. The following is a brief description of how to import and export reports formats. Do not confuse this with exporting data resulting from a report.

Caution: If you have customized the standard reports in ExpressMaintenance, importing a report template file will overwrite your existing reports. You should save any customized reports to a new name before importing report template files.

**Exporting Report Template (Formats)**

1. Run the application such as ExpressMaintenance.
2. Open the applicable report screen. Example: Maintenance / Reports.
3. Locate the desired report in the list of reports in the right panel.
4. Right click on the report and the popup menu will appear.
5. Left click on the "Export Select Report to File" option.
6. Select the desired destination folder where the report will be exported using the report name with the .rtm (or .rdf*) extension.
7. Email the .rtm file as an attachment or keep it as a backup.

**Importing Report Template Files (Formats)**

1. Locate the desired report template file and download - click here for list. Note: Some browsers require that you right click on the desired file in order to download.
2. Or, if emailed, save the report (.rtm) file to disk, noting the folder where the file is saved.
3. Run the application such as ExpressMaintenance.
5. Right click on any report and the popup menu will appear.
6. Left click on the "Import Report(s) from File" option.
7. Use the windows dialog box to locate the report template (.rtm) file.
8. Click the Open button to open the file.
9. The report will be imported and a dialog will reflect the completion.

*Note: The file types for some applications which use the Shazam Report Designer is *.rdf (report definition file). Applications are being converted to the Report Builder which use *.rtm files.

You can download original report definitions for all reports - click here for list.

**1.2.2 Add Fields**

**How do I add fields or adjust the layout of a report?**

ExpressMaintenance includes a very powerful built-in report builder. All of the standard reports included with ExpressMaintenance are built using the report builder. It is often desirable to adjust reports to include or exclude certain fields. While you can develop new reports from scratch, it is usually easiest to edit an existing report that is close to the desired results.

This example will show how to alter a simple report. It provides some introductory concepts of using the report builder. More advance uses of the report builder are certainly possible. However, it is best to have some knowledge of database and report design concepts.

You should make sure your toolbars are Turned-On in the Report Builder. You can do this under View / Toolbars. We recommend turning everything but Data Tree and Data Tree. You can also arrange the
tool bars by dragging them around in the window.

The report used in this example is the Units List By Location report. We will add the Unit number field to the report. Adding a field to a report involves two primary steps. All reports are made up of two building blocks which produce the report that is previewed and printed. These include:

1. Query to extract the desired data
2. Layout / design presentation of the data

First, the field must be included in the query (search) and be included as part of the returned results. Secondly, the data field must be included in the report design to appear on the report.

**Add Data Field To The Query**

1. Run the Unit List By Location report to preview on the screen.
2. Click on the Data tab of the report builder.
3. The Units table will appear as a box in the upper left portion of the screen.
4. The Units window includes several buttons that make up the Query Designer.
5. Click on the Fields button to open the Query Designer to the Fields tab.
6. The upper section reflects the available fields and lower section reflects selected fields.
7. Add the "Unit" (for example) field to the report query by locating the "Unit" field in the upper section and double click on it. This adds the Unit field to the bottom portion of the screen.
8. Edit the query filter criteria using the operator and value desired. When using Between, separate values with a comma.
9. Check AutoSearch to make the query dynamic, meaning it can be changed each time the report is executed. Check mandatory to require the filter each time.
10. Click the Ok button to save the changes in the Query Designer.
11. You can view the raw query results by clicking on the Preview button in the Units window.
12. To learn more about the Query Designer, review the help under Report Builder.

**Add Field To The Layout**

1. Click on the Design tab of the report designer.
2. Locate the Data Components toolbar which is on the top row by default. The first button appears to have an "A" in front of a table. When pointing at the button, the popup hint reads "DBText". This component is for displaying standard data fields. If adding a memo field, you will want to use the DBRichText component follow these steps plus notes below.
3. Click the DBText button and then click in the Detail band of the report layout. A DBText component will be place in that location.
4. Click the Place tab in the Page Designer dialog.
5. Size and adjust the DBText component as desired.
6. Left click on the DBText component and then look in the upper section to locate the table Data Pipeline and Data Field drop down selection fields.
7. Select the Units Data Pipeline if not already selected.
8. For the Data Field, drop down the list and select the desired field such as Unit. This associates the DBText component with the Unit number field.
9. Click the Preview to view the results.
10. To add a label for the Unit field, click on the Label component and place it in the header band. In the upper section, edit the caption of the label to read "Unit Number" or other desired value.
11. Click the Preview tab to view the results.
12. On exit of the report, save your changes.

**Other Notes**

1. If you are placing a notes field in the report, use a DBRichText component rather than a DBText component. You will also need to set the band to dynamic height and the component to stretch.
2. If you are placing a graphic field in the report, use a DBImage component rather than a DBText component. You will also need to set the band to dynamic height and the component to stretch.
3. For more comprehensive help on the report builder, view the help or other Tech Bulletins.

You can download original report definitions for all reports - click here for list.

1.2.3 Receive Filters

How can I make my report receive filters from ExpressMaintenance?

When a report is run in ExpressMaintenance, it first clears all filters in the report unless they are marked as mandatory. Next, the program checks the filters selected in the report filters panel (example: Type or Category). For each filter that contains a value other than "<All>" , the program attempts to pass the filter to the report. If filtering is successful, ExpressMaintenance also populates the "Params" text field in the report if it exist.

You should make sure your toolbars are Turned-On in the Report Builder. You can do this under View / Toolbars. We recommend turning everything but Data Tree and Data Tree. You can also arrange the tool bars by dragging them around in the window.

The program can only pass filters to fields that exist in the tables being used in the Data tab of the report. The field does not have to be in the query result set but it does have to be part of the table. In order for the filter to be applied, the corresponding field name must exist in the table(s) that are in the query but does not have to be used in the Layout.

To view the available field in a query, follow these steps:
1. Run the Unit List By Location report to preview on the screen.
2. Click on the Data tab of the report builder.
3. The Units table will appear as a box in the upper left portion of the screen.
4. The Units window includes several buttons that make up the Query Designer.
5. Click on the Fields button to open the Query Designer to the Fields tab.
6. The upper section reflects the available fields and lower section reflects selected fields.
7. Add the "Unit" (for example) field to the report query by locating the "Unit" field in the upper section and double click on it. This adds the Unit field to the bottom portion of the screen.
8. Edit the query filter criteria using the operator and value desired. When using Between, separate values with a comma.
9. Check AutoSearch to make the query dynamic, meaning it can be changed each time the report is executed. Check mandatory to require the filter each time.
10. Click the Ok button to save the changes in the Query Designer.
11. You can view the raw query results by clicking on the Preview button in the Units window.
12. To learn more about the Query Designer, review the help under Report Builder.

The Maintenance / Reports Screen passes filters to the following field names:

<table>
<thead>
<tr>
<th>Screen Prompt</th>
<th>Table</th>
<th>Field Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed Date</td>
<td>WoMaster</td>
<td>CompletedDate</td>
</tr>
<tr>
<td>Scheduled Date</td>
<td>WoMaster</td>
<td>ScheduledDate</td>
</tr>
<tr>
<td>Performed Date</td>
<td>Servhist</td>
<td>Performed</td>
</tr>
<tr>
<td>Downtime Date</td>
<td>Downtime</td>
<td>DateDown</td>
</tr>
<tr>
<td>Unit Name</td>
<td>Units</td>
<td>Name</td>
</tr>
<tr>
<td>Unit Number</td>
<td>Units</td>
<td>Unit</td>
</tr>
<tr>
<td>Site</td>
<td>Units / WoMaster</td>
<td>Site</td>
</tr>
<tr>
<td>Locations</td>
<td>Units</td>
<td>Location</td>
</tr>
<tr>
<td>Unit Type</td>
<td>Units</td>
<td>Type</td>
</tr>
<tr>
<td>Unit Category</td>
<td>Units</td>
<td>CategoryID</td>
</tr>
<tr>
<td>Part Description</td>
<td>Parts / WoDetail</td>
<td>Description</td>
</tr>
</tbody>
</table>
You can apply your own filters after the report is executed and being viewed. Simply go to the Data tab and click on the Search button. Double click on the desired field and edit the search criteria with the desired operator and value. You can make a filter permanent by selecting the mandatory option.

You can download original report definitions for all reports - [click here for list](#).

### 1.2.4 Adjust Search

**How to make a report include all of the data records I expect?**

You should make sure your toolbars are Turned-On in the Report Builder. You can do this under View / Toolbars. We recommend turning everything but Data Tree and Data Tree. You can also arrange the tool bars by dragging them around in the window.

When a report is empty or does not include the expected data, this meaning it shows limited or no records found. This a direct result of the query being applied to search for the data. The query statement is comprised of the search filters that are entered in the Query under the Data tab of the Report Builder.

**How can I check the data query?**

1. Click on the Data tab of the report builder
2. The query will appear in a window in the upper left section of the screen
3. Locate and click on the Search button (magnify glass)
4. The Query Designer will appear with the Search tab selected
5. Examine the lower section of the screen for the search criteria being applied
6. Adjust the search criteria as desired to get the desired data result set
7. Edit the query filter criteria using the operator and value desired. When using Between, separate values with a comma.
8. Check AutoSearch to make the query dynamic, meaning it can be changed each time the report is executed. Check mandatory to require the filter each time.
9. To make a query filter permanent, click on the mandatory checkbox. To make it permanent and not to be auto overwritten from the program, un-check AutoSearch.
10. Click Ok to save the changes.
11. In the Data tab and query table window, click the Preview button to see the raw data results of the query.

**How do I correct a report with this problem?**

1. One option is to edit the filter as outlined above
2. Another option is to import the original report template by downloading from website page - [click here for list](#). The page also includes a link on instructions for importing report template files.

**What if the program does not allow me to edit reports?**

You must have permission to make report changes in order to adjust the report. If you do not have permission, see your system administrator for assistance.

You can download original report definitions for all reports - [click here for list](#).
1.2.5  More Help

How can I get comprehensive help or a tutorial on Report Builder?

ExpressMaintenance and other products developed by Express Technology include the very powerful Report Builder. The report builder is a third party component that we compile directly into the ExpressMaintenance. It provides a very flexible query / reporting tool that allows you full access to your data with virtually unlimited presentation capabilities.

Report Builder is very easy to use. However, using any report designer does take some time and practice to become efficient. Three additional help tools are available to enhance your use and understanding of the report builder. These include help files on the report builder and SQL as well as a tutorial on the Report Builder.

The Report Builder help file is automatically included in the ExpressMaintenance application help. Check the chapter titled "Report Builder". The help file is also available as an Adobe pdf file.
Click here to download report builder help in pdf manual format

Complete Learning Report Builder Tutorial
Learning Report Builder is a complete learning system designed to teach end users how to build a range of reports. This system includes a 125-page PDF file, a stand-alone application complete with database, and help file. The PDF file is comprised of a series of tutorials that step end users through the process of building reports as simple as a table listing and as complex as crosstabs. The tutorials introduce conceptual aspects of report building along the way. The application is used in conjunction with the tutorials so the learning experience is interactive. Users print out the PDF file, run the application, and learn Report Builder. The help file is accessible from the application, so all three tools work together to provide a seamless learning environment.

Learning Report Builder installs into C:\ProgramFiles\LearnReportBuilder. When the system has successfully installed, a complete set of instructions appears in the form of a 'ReadMe' doc, so the end user knows what to do next.
Click here to download the Learning Report Builder tutorial

The Learning Report Builder PDF is also available as a bound book. For ordering directly from Digital-Metaphors, visit:
https://www.digital-metaphors.com/secure/orderform.html

SQL Help & Concepts
It is important to understand that the report designer is simply a graphical interface for building SQL queries. Therefore, a better understanding of SQL and the Select statement will provide an enhanced understanding of how to query the data for reporting purposes.
Click here to download SQL Books Online.

Because Express Technology software is based on the Microsoft SQL Server database engine, other reporting tools can be used as well. Included are such items as Microsoft Access and Crystal Reports.

You can download original report definitions for all reports - click here for list.
1.3 Workspaces

1.3.1 About the Report Designer

Report Designer

The Report Designer is the environment in which reports are built. The Report Designer is comprised of the following three workspaces, which are accessible by tabs bearing their name:

- Data
- Design
- Preview

1.3.2 Data Workspace

Data Workspace

The data workspace is where you go to set the data for a report. Query-building tools such as the Query Wizard and the Query Designer are accessible from this screen.
1.3.3 Design Workspace

The design workspace is where you lay out reports. After a new report is created and data is set up in the data workspace, access the design workspace to build a report.

You should make sure your toolbars are Turned-On in the Report Builder. You can do this under View / Toolbars. We recommend turning everything but Data Tree and Data Tree. You can also arrange the toolbars by dragging them around in the window.
1.3.4 Preview Workspace

The preview workspace is used to see the printed layout of a report. This workspace is especially useful when building a report. Place a component in band in the design workspace, then preview to see where it prints and how it looks. Click the tab at the top of the Report Designer marked Preview in order to access this screen. It's a good idea to save the report before selecting the preview tab, as the report generates upon preview.
1.4 Data-Specific Tools

1.4.1 Database

Database

A database is a holding tank for data, or raw facts and figures. Databases are comprised of tables, which are collections of data organized into columns and rows, or fields. A report cannot be created without a database.
1.4.2 DataView

The DataView appears upon the completion of a query. It allows you to make changes to the existing query. When you complete a query, the DataView is displayed in the left corner of the data workspace. Here’s an example of a DataView. Place your cursor over the icons to learn more about them.
1.4.3 Data pipeline

Data Pipeline

Data pipelines represent a database table or SQL query and provide a set of data structured as records and fields. All data-aware components need to be assigned to a data pipeline.

1.4.4 SQL

SQL

SQL stands for Structured Query Language. It's a language used to access data from a database. You can create SQL queries without knowing the language via the Query Wizard or the Query Designer. You cannot build a report without accessing data from a database.

1.4.5 Query Wizard

Query Wizard

The Query Wizard is a tool that helps you choose the data you want to use for a report. As you select this data, the Query Wizard builds an SQL statement, which allows you to select data from a database. When you complete a query via the Query Wizard, a DataView is displayed. You can go back into a query and make changes via the DataView. When you re-enter a query, you do so by using the Query Designer. Click the Data tab and select File | New in order to access the Query Wizard.

Select the Table(s) for the query
Select the order of the resulting query

Add any calculated or concatenated fields
Add any grouping to only show totals or sums

Select the tables that you want to query.

- [ ] No Grouping
- [ ] Select Group Fields

Complete and generate the query. Can be edited with the query designer.

1.4.6 Query Designer

The Query Designer is a tool that allows you to select data from a database without using SQL. When you complete a query, a DataView is displayed. Click the Data tab and select File | New in order to access the Query Designer.

Select table(s) to be used in the query
Select fields to be in the query

Create any calculated fields
Define any grouping fields for total or summary reports

Define any search criteria for the query
Define the sort order for the query results

View the resulting SQL statement
1.5 Report-Building Tools

1.5.1 About report-building tools

Report-Building Tools

The Report Designer offers three tools that generate reports for you based on the selections you make when working with them. These tools, or wizards, are available in the design workspace. Select File | New, then click on the icon to access a wizard.
1.5.2  Report Wizard

Report Wizard

The Report Wizard is a tool in the design workspace that allows you to create a report by answering some questions pertaining to data, layout, and style. Access this tool by selecting File | New, then clicking the Report Wizard icon.

Select Field To Be Included
Define Group If Desired
Select Grouping Orientation
Select Report Style
Finish The Report
Report Designer Help

That is all the information needed to create your report.

Do you want to preview the report or modify the report's design?

- [ ] Preview the report
- [ ] Modify the report's design

Report Builder Creates the Report which you can then edit and adjust.
1.5.3 Label Template Wizard

Label Template Wizard

The Label Template Wizard is a tool in the design workspace that allows you to build templates by selecting printer information and choosing a label type. Select File | New in order to access this tool.
1.5.4 CrossTab Wizard

The CrossTab Wizard is a tool in the design workspace that generates a report based upon the choices you make in the wizard. You can further configure the crosstab component by right-clicking and selecting Configure. Access this tool by selecting File > New, then clicking the CrossTab Wizard icon.
1.6 Toolbars

1.6.1 About toolbars

About toolbars

The design workspace has several toolbars that are used in order to create, adjust, or modify components. Toolbars can be launched by selecting View | Toolbars.

You should make sure your toolbars are Turned-On in the Report Builder. You can do this under View / Toolbars. We recommend turning everything but Data Tree and Data Tree. You can also arrange the toolbars by dragging them around in the window.

1.6.2 Advanced Component Palette

Advanced Component Palette

The Advanced component palette contains components that can help you tackle complex reporting requirements. Place your cursor over the icons to learn about the components.
1.6.3 **Align or Space Toolbar**

**Align or Space Toolbar**

This toolbar is useful when components need to be positioned uniformly. For example, it can align several components so that the tops are all even, or it can space components so that they have an equal amount of space between them. The first component selected determines the position to which the others will align.

1.6.4 **Data Component Palette**

**Data Component Palette**

This palette offers several components that are similar to components in the standard palette, except for one thing: they are data-aware. Data-aware components can read the value of a database field. Associate a data-aware component with a given table and field in the database by selecting the data pipeline and the data field from the drop-down lists in the **Edit toolbar**. Place your cursor over the icons to learn about the components.

1.6.5 **Data Tree**

**Data Tree**
The Data Tree can be used to create data-aware components within a band. Simply select a set of fields and drag the selection into the band. A set of corresponding data-aware components will be created. This tool window is dockable only on the left and right sides of the design workspace.

1.6.6 Draw Toolbar

Draw Toolbar

This toolbar is used to set the color and style for the shape, line, and region components.

1.6.7 Standard Toolbar

Standard Toolbar

The Standard toolbar allows you to perform basic functions such as opening, saving, and printing reports, as well as cutting and pasting selections.

1.6.8 Edit Toolbar

Edit Toolbar

This toolbar changes according to the component that is selected. Place your cursor over the fields to learn more about them.
**Edit toolbar with a data-aware component**

This configuration allows the data pipeline and data field for the component to be set. The drop-down list on the left shows the data pipelines. The drop-down list on the right shows the field names.

**Edit toolbar with a label component**

This configuration allows you to type the text for a label.

**Edit toolbar with a shape component**
The Edit toolbar allows you to choose from several shapes when a shape component is selected.

**Edit toolbar with a line component**

![Edit toolbar with a line component]

This configuration allows you to move the line to the top, bottom, left, or right within a line component.

1.6.9 **Format Toolbar**

**Format Toolbar**

This toolbar sets the font and colors of text-based components. It also sets the layering of all components.

![Format Toolbar]

1.6.10 **Nudge Toolbar**

**Nudge Toolbar**

This toolbar is useful when you want to move a component or selection of components with extreme precision. Each icon represents the direction the selection will move. Selections will move one pixel each time you press an arrow key.

![Nudge Toolbar]

1.6.11 **Report Tree**

**Report Tree**

The Report Tree can be used to view the components within each band. Components selected in the Report Tree become the selection in the report layout. You can select multiple components by holding down the Ctrl key and clicking on each name. You can rename components by right-clicking over the name, selecting Rename, and then typing in a new name. Make sure to hit the Enter key after renaming to ensure that the new name is assigned. This tool window is dockable only on the left and right sides of the design workspace.
1.6.12 Standard Component Palette

Standard Component Palette

The icons on the Standard component palette represent components that are frequently used to build reports. To create a component, click on an icon and then click in a band. Place your cursor over the icons to learn about the components.

1.7 Bands

1.7.1 About Bands

About Bands

A band is a section of the design workspace that describes how parts of the report will look. Bands are labeled in the section divider immediately below them; thus, the first band is called the 'header', the second is called the 'detail', and the third is called the 'footer.' The header, detail, and footer bands appear in this order as part of the layout of a newly-created report.
### 1.7.2 Header

#### Header Band

The header band prints at the top of each page. You can remove a header band from the layout by selecting Report, then clicking on Header.
1.7.3 Group

A group is a section of a report that contains a group header, detail, and group footer band. Groups are assigned to a database field. Select Report | Groups to access the groups dialog. If you want the groups to print on separate pages, select the Keep group together option in the groups dialog.

1.7.4 Group Header

The group header band contains the header for a group. The components in a group header band appear at the beginning of a group. Select Report | Groups to create a group header and footer band.
1.7.5 Detail

Detail Band

The detail band prints once for every row of data.
1.7.6 Footer

Footer Band

The footer band prints its components (usually System Variables) on the bottom of each page. To remove this band from the layout, select Report, then click on Footer.

1.7.7 Group Footer

Group Footer Band

The group footer band contains the footer for a group. The components in this band print at the end of each group; therefore, if there are several groups per page, the contents of this band will print several times. Select Report | Groups to create a group footer and header band.
1.7.8 Summary

Summary Band

The summary band prints once at the end of a report. The band is typically used to summarize data. Select Report | Summary to create this band.
1.7.9 Title

Title Band

The title band contains the title of the report. It prints on the first page only. Select Report | Title to create this band.

1.8 Components

1.8.1 Advanced

1.8.1.1 CrossTab

CrossTab

The CrossTab is a component on the Advanced component palette that allows you to generate a set of calculations that summarize the data from a database table. It displays the calculations in a grid format. Right-click over the component and select Configure to set up the crosstab.

1.8.1.2 Region

Region

A region is a component on the Advanced component palette that can contain other components. To print a region after a text-printing component (memo, DBMemo, RichText, DBRichText), right-click and select ShiftRelativeTo.
1.8.3 SubReport

SubReport

The SubReport is a component on the Advanced component palette that allows you to create a report within a report in order to show more levels of detail or to print several reports as one.

1.8.2 Standard

1.8.2.1 BarCode

BarCode is a component in the Standard component palette that renders barcodes. Use the Edit toolbar to set the data to be encoded. Right-click and access Configure to pick bar code types.
1.8.2.2 Checkbox

Checkbox is a component in the Standard component palette that renders checkboxes based on yes / no or true / false values. Use the Edit toolbar to set the data to be reflected. Right-click and access the Checked property.

1.8.2.3 Image

Image is a component on the Standard component palette that displays graphics (such as bitmaps, GIFs, and JPEGs). Right-click over the component and select MaintainAspectRatio to scale an image so that it is the same height and width.

1.8.2.4 Label

A label is a component in the Standard component palette that functions as a header for another kind of component, such as a DBText. Use the Edit toolbar to type text into the label.
1.8.2.5 Line

Line
A line is a component in the Standard component palette that displays a line. Use the Edit toolbar to set the line orientation.

1.8.2.6 Memo

Memo
A memo is a component on the Standard component palette that allows you to load multiple lines of plain text into it. Right-click over the memo component and select Lines, then click the Load option to bring text into a memo. To print a memo after another text-printing component (DBMemo, RichText, DBRichText), right-click and select ShiftRelativeTo.

1.8.2.7 RichText

RichText
RichText is a component in the Standard component palette that prints formatted text. Right-click over the component and select MailMerge to bring a file into the component. To print a RichText after another text-printing component (memo, DBMemo, DBRichText), right-click and select ShiftRelativeTo.

1.8.2.8 System Variable

System Variable
The System Variable is a component on the Standard component palette that displays common report information such as the page number, time, and date.

1.8.2.9 Shape

Shape
A shape is a component in the Standard component palette that displays various shapes, such as squares, rectangles, circles, and ellipses. Use the Edit toolbar to change the shape type.

1.8.2.10 Variable

Variable
A variable is a component in the Standard component palette that performs calculations.
1.8.3 Data-aware

1.8.3.1 DBBarCode

**DBBarCode**
The DBBarCode is a component on the Data component palette that converts the data from a database field into a barcode symbol.

1.8.3.2 DBCalc

**DBCalc**
The DBCalc is a component on the Data component palette that performs simple database calculations (Sum, Min, Max, Average, Count).

1.8.3.3 DBCheckBox

**DBCheckBox**
The DBCheckBox is a component on the Data component palette that performs a display of a checked or unchecked box depending on data value. Data must be of a True / False or Yes / No (one of two values) to function properly in a checkbox. Right click on the placed component to set the properties.

1.8.3.4 DBImage

**DBImage**
The DBImage is a component on the Data component palette that prints graphics (Bitmaps, GIFs, JPEGs) that are stored in a database field. Right-click over the component and select MaintainAspectRatio to scale an image so that it is the same height and width.

1.8.3.5 DBMemo

**DBMemo**
The DBMemo is a component on the Data component palette that prints plain text from a memo field of a database table. It will automatically word-wrap the text. Select the DBMemo, then select a text-based field from the Edit toolbar to fill a DBMemo with text. To print a DBMemo after another text-printing component (memo, RichText, DBRichText), right-click and select ShiftRelativeTo.

1.8.3.6 DBRichText

**DBRichText**
The DBRichText is a component on the Data component palette that prints formatted text from a
memo field. It will automatically word-wrap the text. To print a DBRichText after another text-
printing component (memo, DBMemo, RichText), right-click and select ShiftRelativeTo.

1.8.3.7 DBText

![DBText](image)

The DBText displays data from most types of database fields. It cannot handle images or Rich Text.

1.8.4 Speed Menu Options

1.8.4.1 AutoSize

AutoSize

AutoSize is a speed menu option available for text components. When AutoSize is set to true, the
component adjusts its width so that all of the text is displayed.

1.8.4.2 Bring to Front

Bring To Front

Bring to Front allows you to move an object in front of other objects. Use this option to control the
appearance of overlapping objects.

1.8.4.3 Configure

Configure

Configure is a speed menu option for the Crosstab component. It allows you to access the Crosstab
Designer.

1.8.4.4 DisplayFormat

DisplayFormat

DisplayFormat is a speed menu option available for textual components. This option allows you to
display a value in a certain format (decimals with a dollar sign, for example). Right-click over a
textual component to access the Display Format.

1.8.4.5 Edit

Edit

Edit is a speed menu option for a Rich Text component that opens a dialog which allows you to open
a file and format text.

1.8.4.6 Lines

Lines

This memo speed menu option that allows you to access the Memo Editor, from which you can open
text only files.
1.8.4.7 **MailMerge**

**MailMerge**

MailMerge appears when you right-click over a Rich Text component. It allows you to import field values into the Rich Text component.

1.8.4.8 **MaintainAspectRatio**

**MaintainAspectRatio**

MaintainAspectRatio is a speed menu option available for images. It scales an image so that the height and width are adjusted based on the original image size.

1.8.4.9 **Pagination**

**Pagination**

Pagination is a speed menu option for the crosstab component that allows you to choose Across then Down or Down then Across printing for the crosstab report.

1.8.4.10 **ParentHeight**

**ParentHeight**

ParentHeight is a speed menu option that allows you to set the shape of an object so that it matches the height of a band. Right-click over a component to select ParentHeight.

1.8.4.11 **ParentWidth**

**ParentWidth**

ParentWidth allows you to set the shape of an object so that it matches the width of a band. Right-click over a component to select ParentWidth.

1.8.4.12 **Position**

**Position**

Position is a speed menu dialog that allows you to change size of an object. Right-click over a component to access this dialog.

1.8.4.13 **ReprintOnOverFlow**

**ReprintOnOverFlow**

ReprintOnOverFlow is a speed menu option that applies to static components, or components that always print at the same height. These types of components are frequently used with stretchable components because they serve the purpose of a heading. If a memo prints onto several pages, the heading component should print with it. Set the heading component to ReprintOnOverFlow so it will print on each page.
Send to Back is frequently used with shapes. This speed menu option allows components to be moved to the background. If you want to use a shape as a background, use the Send to Back function.

ShiftRelativeTo is a speed menu option available to the following components:

- Memo
- DBMemo
- RichText
- DBRichText
- Region

This option allows us to associate these components with one another so that one can print directly after another. To make this association, right-click over a component, select ShiftRelativeTo, and choose the component you want to print first from the drop-down list.

StretchWithParent allows the selected component to change size according to the change in height of the band in which it resides. Right-click over a component to access this option.

ShiftWithParent allows a static component to print after a stretchable component. Right-click over a component to select ShiftWithParent.

This menu option applies to stretchable components only. When Stretch is not selected, the component behaves just like a static component: it prints the same height each time, regardless of text. When Stretch is selected, the component adjusts according to the amount of printing text.

Style is a speed menu option for the crosstab component that allows you to choose between Repeated Captions or Standard printing.
1.8.4.20 **SuppressRepeatedValues**

**SuppressRepeatedValues**

This speed menu option prevents repeated values from printing for DBText and DBCalc components.

1.8.4.21 **Visible**

**Visible**

This speed menu option determines the visibility of an object. To set the visibility of a component to False, right-click over it and deselect Visible.

1.8.5 **Static VS. Stretchable**

1.8.5.1 **Stretchable**

**stretchable**

Stretchable components print according to the amount of text they contain; therefore, the size changes depending upon their contents. A component is only stretchable if it can contain several lines of text. For example, the Memo, DBMemo, RichText, DBRichText, and Region components are all stretchable.

1.8.5.2 **Static**

**static components**

Static components always print at the same height: they do not stretch to accommodate the length of text. They do, however, adjust their width to fit long words if the AutoSize option is selected. The label, DBText, and System Variable are all static components.

1.9 **Reference**

1.9.1 **Glossary**

**A**

**Advanced component palette**

align

Align or Space toolbar

AutoSize

**B**

bands
BarCode
bounding box
BringToFront

C

canvas

components
Configure
CrossTab
CrossTab Designer
CrossTab Wizard

D
data
data-aware
database
database table
Data component palette
data module
data pipeline
data traversal
Data Tree
data workspace
DataView
DBBarCode
DBCalc
DBImage
DBMemo
DBRichText
DBText
design workspace
detail band
DisplayFormat
dock
Draw toolbar

E
Edit
Edit toolbar

F
fields
floating window
footer band
Format toolbar

G
group
group footer band
group header band
guides

H
header band
highlight
I
Image

J

K
Keep group together

L
Label
Label Template Wizard
Line
Lines

M
MailMerge
MaintainAspectRatio
Memo
Memo Editor

N
Nudge toolbar

O
orphan
overflow

P
Pagination
ParentHeight
ParentWidth
Position
preview screen
properties
Q
query
Query Designer
Query Wizard
R
Region
report
Report Designer
Report Tree
Report Wizard
ReprintOnOverFlow
RichText
RichText Editor
rulers
S
select
selection
selection handles
Selection Tool
Send to Back
Shape
shift-click
ShiftRelativeTo
ShiftWithParent
Size toolbar
sizing
sizing handles
speed menu
SQL
Standard component palette
Standard toolbar
Start new page
static
status bar
stretch
stretchable components
StretchWithParent
Style
SubReport
Summary band
SuppressRepeatedValues
System Variable
T

tabular

title band

title bar

U
1.10  RAP Reference

1.10.1  What is RAP

1.10.1.1  Overview of Features

**RAP allows you to store code with your reports.** Until now, ReportBuilder's powerful events have been available only within Delphi. If you wanted to load reports at runtime and retain any event handlers, you had to load the report into a form or datamodule which contained correctly named procedures in order to ensure that the loaded report "hooked up" successfully to the event handlers. Now, with RAP, you can code your event handlers within the **Calc workspace** and then save them as part of the report in an RTM file. When you load a report from an RTM or from a database, your event handlers are loaded as well and are already "hooked up".

**RAP allows your users to create their own calculations with their reports.** ReportBuilder's award winning end user solution was already powerful, allowing your users to edit or create new reports, but RAP extends the solution by giving end users the ability to code their own event handlers and extended calculations. The Calc workspace is an Object Pascal development environment that is designed for ease of use by non-developers. The **Code Explorer** offers varied views of the report code module; the **Code Editor** is a syntax-sensitive Pascal editor; the **Code Toolbox** serves double duty, providing both a partial list of supported identifiers as well as a drag & drop code creation facility.

**RAP is scalable.** ReportBuilder allows you to **scale RAP** to the needs of your users. The Calc workspace has a great deal of functionality, but your users may not need all of it. You are able to limit RAP to those parts you wish to deploy. Using the RAPIface and RAPOptions properties of the Designer component, you can specify what features to make available to the end user.

**RAP is a subset of Object Pascal** so you already know Report Application Pascal (RAP) and can easily support your users.

**RAP is extensible.** RAP is installed with a large number of standard Delphi functions, Delphi objects and RCL (Report Component Library) objects already registered with the compiler and Code Toolbox.
However, if you wish to register your own functions or objects, it is an easy process. Thus you can build a library of pass-through functions to deploy with your report applications.

**RAP provides a simple, intuitive Pascal development environment.** While you can use the Calc tab to add code to your reports, it is also made for the end user who may or may not have any experience in development environments. See the Calc workspace for information about the IDE.

### 1.10.1.2 Overview of the Interface

#### 1.10.1.2.1 The Calc Tab

The Calc workspace is the development environment for RAP. You can include the Calc workspace in your end user projects by adding raIDE to the uses clause of one of your units.

When your users select the Calc tab, they will see the RAP IDE:

The RAP IDE consists of

- **The Code Explorer**
- **The Code Editor**
- **The Code Toolbox**
1.10.1.2.2 The Code Explorer

The Code Explorer is contained in the upper left and right panes of the Calc workspace.

The left pane contains a tree view — use this to navigate your report's code. The right pane contains a list view — it will display a variety of items depending on what is selected in the tree view.

By right clicking on the tree you can display a context menu that allows you to control the behavior of the Code Explorer.

If you select Variables, the tree will display the bands of the report. When you click on a band, all variables contained in the band will be displayed. Selecting an individual variable causes the Code Editor to display the OnCalc event handler, if one exists.

If you select Events, the tree will display all components in the report. Selecting an individual component allows you to see all of the events for that component. When you select an event, the Code Editor will display the event handler, if one exists.

If you select Module, the tree will display module level items: Declarations, Events, Programs (procedures and functions) and Event Handlers. The items displayed in the list view will depend on what is selected in the tree. Likewise, the Code Editor will display code appropriate to what is selected in the list view.

With Declarations selected in the tree, the list view will show Constants and Variables. Select either item to enter declarations in the Code Editor.

Message Window.
With Events selected in the tree, the Module level events (OnCreate and OnDestroy) will be shown. Select either to edit the code for these events. These events correspond to a TppReport’s BeforePrint and AfterPrint events.

With Programs selected in the tree, the list view lists any Module level procedures or functions you have defined. Use the Code Editor to edit these programs.

With Event Handlers selected in the tree, the list view displays any events in the report that have handlers assigned. Use the Code Editor to edit these handlers.

1.10.1.2.3 The Code Editor

The Code Editor is the place where you actually write and modify RAP code.

This syntax-sensitive editor is similar to the one found in Delphi, with the exception that it displays only one procedure at a time. Whatever the currently selected item is (event handler, procedure, function, variable declarations, etc.), only the code for that item is displayed in the Code Editor.

With an item is selected, the editor will either contain the code implementation, or will be blank (if no implementation exists). If no implementation exists, you can create one by clicking in the editor. You can then enter code by either typing or by dragging items from the Code Toolbox and dropping them into the editor. If an item is dropped into the editor over a section of selected code, the selected code will be replaced.

The Code Editor’s context menu contains the following items:
New
New has the same effect as clicking in the Code Editor. It is only enabled if there is no implementation for the item currently selected in the Code Explorer.

Compile
Compile activates the RAP compiler to attempt to compile the current procedure and any procedures upon which the current one depends.

Save
The Calc workspace maintains an intermediate buffer for the Code Editor. Selecting Save will commit the current contents of the Code Editor to the buffer; it will not save the entire report. Selecting Save has the same effect as navigating away from, and then returning to the current procedure.

Revert
Use Revert to replace the contents of the Code Editor with what is currently contained in the code buffer. This has the effect of removing all changes since the last save.

Delete
Select Delete to remove the current procedure entirely.

1.10.1.2.4 The Code Toolbox

The Code Toolbox is a visual code repository. It contains most of the identifiers and code elements that the RAP compiler recognizes.

The Code Toolbox enables you to:

- View identifiers grouped by available data, visible object properties or language features.
- Generate code by dragging identifiers into the Code Editor. Functions dragged into the editor will generate a place-holder parameter list. For example, the following code is generated when you drag the Copy function into the Code Editor:

```c
Copy(S, Index, Count);
```

Each tab of the Code Toolbox consists of a treeview and a list of identifiers. The treeview allows you to navigate groups of identifiers listed on each tab.

Identifiers listed in the Toolbox will also display relevant information such as Type, Size, Signature, etc.
Identifiers which appear in the Code Toolbox are registered with RAP via RAP's extensible architecture. You can use this architecture to register your own pass-through functions and objects with RAP — this allows your new functions to appear in the Code Toolbox and to be recognized by the compiler.

1.10.1.2.5 The Message Window

The Message Window functions in essentially the same manner as Delphi's message window. Messages from the compiler are presented here. You can navigate to the location of compiler errors by double-clicking the error message.

1.10.1.2.6 The Variables View

The Variables view of the Code Explorer is displayed by right-clicking the left pane and selecting Variables from the context menu.

This view displays the bands of the report and any variables contained in the currently selected band. Selecting a variable displays the OnCalc event handler, if one exists. The OnCalc handler is presented without the signature, as:

\[ \text{Value := } \]

This is the simplest view available and is good for limiting the Calc workspace to a robust calculations editor.

For an explanation of the triangular icons which appear on the treeview, see the Events View.

1.10.1.2.7 The Events View

The Events view of the Code Explorer is displayed by right-clicking the left pane and selecting Events from the context menu.

This view displays a listing of all components contained within the report. The right pane displays any events associated with the currently selected component. Selecting an event will display the event handler, if one exists.

This view is good for viewing all report objects and their events.
In the treeview, you will notice small arrow shaped images to the left of some nodes. These are Compilation State Indicators. They are used to tell you where your code is and what state it is in. There are five possible indicators:

- **No symbol.** Indicates that this component does not have any event handlers assigned, nor does it contain any components which have event handlers.

- **Indicates that this component does not have any event handlers, but contains components which do.** The red color indicates that one or more of the nested components has event handlers which do not compile.

- **Indicates that this component contains event handlers and that somewhere on this branch there is code that does not compile.** If there is no code contained in components below this one, then the problem code is in this component. However, if there is code below this component, the problem may be in a child component's event handlers, in this component's event handlers or both. Note: If a child component's code does not compile, the parent component will still display a red arrow even though its own code may compile.

- **Indicates that this component does not have any event handlers assigned, but contains other components which do.** The green color indicates that the event handlers of the contained components have compiled successfully.

- **Indicates that this component has event handlers assigned.** The green color indicates that these event handlers, and any assigned to components nested within this one, have successfully compiled.

1.10.1.2.8 The Module View

The Module view of the [Code Explorer](#) is displayed by right-clicking the left pane and selecting Module from the context menu.
This view displays items which are visible to all event handlers of the report:

Declarations – These are variables and constants that are globally visible throughout the report.

Events – These are, in essence, the report’s events. In the case where the preview window is displayed, OnCreate and OnDestroy fire when the window is opened and closed, respectively. This is different from Report.BeforePrint and AfterPrint in that those methods will fire each time Report.Print is called. OnCreate and OnDestroy are good places for initialization and finalization code such as creating and freeing objects and initializing variables.
Programs – These are procedures and functions that are globally visible throughout the report and can therefore be called from any event handler.

Event Handlers – These are all event handlers that have been implemented in the report.

See this note on the term "Global".

For an explanation of the triangular icons which appear in the tree view, see the Events View.

1.10.1.2.9 Context-Sensitive Help

Context-sensitive help can be accessed from anywhere in the Calc workspace, simply by clicking the F1 key. The context is determined by the currently focused control.

In the Code Explorer, if the treeview has focus, then help will be displayed for the currently selected object.
If the Code Explorer's listview has focus, help will be displayed for the selected event.

In Module view, help will be displayed about the selected Global object.

In the Code Editor, the context is determined by the current position of the cursor.

In the Code Toolbox, the displayed topic is based on the selected pipeline, field, object, property, method or language item.
In the Message window, if an error is highlighted, a topic concerning that error will be displayed.

1.10.2 Quick Start Tutorials

1.10.2.1 Color-coding a DBText Component

In this tutorial, we will have a quick look at using RAP at run-time.

Note: This tutorial assumes you have installed both ReportBuilder Pro and RAP (or ReportBuilder Enterprise).

If it is not already running, start Delphi and load the ReportBuilder end user demo project, RBuilder\Demos\3. EndUser\1. Report Explorer\EndUser.dpr.

Enable RAP by removing the "x" from in front of $DEFINE RAP in the MyEUrpt.pas interface.

Compile and run the demo project.

1.10.2.2 Concatenating Fields

This tutorial will walk you through a basic report created at design-time. We will create a report that concatenates two fields using RAP instead of Delphi event handlers.

Note: This tutorial assumes you have installed both ReportBuilder Enterprise and the RAP design-time package.

Click Next to begin.

1.10.2.3 Dynamic Duplexing

Problem: "I want to use duplexing to print a disclaimer on the back of every page of my report."

Ok, that's easy.

Basically we need to cause the data to print only on even numbered pages and the disclaimer to print only on odd numbered pages.

At runtime, we'll create a report layout to address the problem, then write some RAP code to manage the solution.

Note: This tutorial assumes you have installed ReportBuilder Enterprise.
1.10.2.4 Adding New Functions to RAP

This tutorial will walk you through adding two functions and a category to the functions list in the Code Toolbox.
We will be adding a pass-through function to retrieve the application's filename, a function to expose the ExtractFilename Delphi function, and a category named "Filename".

To do this we will create a new unit to contain the new functions, enable RAP in the End User demo, add the new unit to that demo then run the demo to see the new functions displayed.

**Note:** This tutorial assumes you have installed both ReportBuilder Enterprise and the RAP design-time package installed.

1.10.2.5 Extending the RAP RTTI

This tutorial will walk you through the process of making RAP aware of a new component. We will be registering a new class to surface TDataBase within RAP. In addition, we will add support for the public property, Directory and the public method, ValidateName.

To do this we will create a new unit to contain the new class, enable RAP in the End User demo, add the new unit to that demo, then run the demo to see the new functions displayed.

**Note:** This tutorial assumes you have installed both ReportBuilder Enterprise and the RAP design-time package installed.

**Note:** This tutorial duplicates code found in the RAP demo in `RBuilder\Demos\0.RAP\myRapFuncs0034.pas`.

1.10.2.6 Printing a Description of AutoSearch Criteria

This tutorial will walk you through gaining access to the AutoSearch field descriptions via RAP. This is a simple process and will allow you to display values the user has specified for their AutoSearch criteria.

We will be creating a new report and dataview for this tutorial.

**Note:** This tutorial assumes you have installed both ReportBuilder Enterprise and the RAP design-time package.

1.10.2.7 Displaying Delphi Forms From RAP

The AutoSearch functionality is well suited to polling the end user for search criteria, however sometimes it is necessary to ask the end user for information that is not related to the database. In Delphi this would be easy, but the end user, without Delphi, would be unable to show custom forms unless you expose them via pass-through functions.

This tutorial will walk you through the process of making custom forms available to the end user in RAP. We will create custom forms and then make them available from within RAP. Then we will create a report which accesses these forms.

**Note:** This tutorial assumes you have both ReportBuilder Enterprise and the RAP design-time package installed.
1.10.3 Programming with RAP

1.10.3.1 Procedures and Functions

1.10.3.1.1 Declaring Local Variables

Local variable declarations in RAP are just the same as in Object Pascal. To add a local variable declaration to a function, activate the Code Editor for the current item and place the cursor between the function's declaration and the `begin`. Type `var` and declare your variables just as you would in Delphi.

1.10.3.1.2 Declaring Local Constants

Local constant declarations in RAP are just the same as in Object Pascal. To add a local constant declaration to a function, activate the Code Editor for the current item and place the cursor between the function's declaration and the `begin`. Type `const` and declare your constants just as you would in Delphi.

1.10.3.1.3 Calling Procedures and Functions

Calling a function in RAP is no different than in Object Pascal. As long as the function is in scope, you can call it.

Note: When calling a parent report's global function from within a subreport, you do not need to qualify the identifier with the parent report's name.

In other words, if your Main report has a global procedure, `GlobalDoSomething`, and you want to call it from within SubReportA's OnCreate event, you do not have to say `Main.GlobalDoSomething`. Merely calling `GlobalDoSomething` will suffice.

1.10.3.1.4 Procedure and Function Parameters

Function parameter lists are the same in RAP as in Object Pascal.

Note: Because event handlers in RAP cannot apply to more than one event, event handlers in RAP do not have a Sender parameter in their signature.

1.10.3.2 Events

1.10.3.2.1 Coding an Event Handler

In order to code a new event handler, simple select an event in the Code Explorer and click in the white space of the Code Editor. Both the signature and the begin/end pair for the event handler will be generated automatically. You can then begin entering your code. A typical event handler would appear as:

```
procedure Variable1OnCalc(var Value: Variant);
begin
end;
```

Note: When you are working in the Variables view of the Code Explorer, the event handler signature and begin/end pair will not appear in the Code Editor. You will see only a single line for assigning the Value:

```
Value :=
```

In this mode the Code Explorer displays only variables and is, in a sense, equating variables with the value returned by the OnCalc event. This is valuable for limiting the amount of functionality you wish to reveal to your users.
1.10.3.2.2 Compiling Event Handlers

The RAP compiler attempts compilation of the entire module automatically when any of the following happen:

- You load a report.
- You select the Calc tab within the Report Designer.
- You switch views in the Code Explorer.

It is also possible to compile the currently selected event handler. To do this, right-click over the Code Editor and select Compile from the popup menu. The compiler will check the global sections and any other programs needed to compile the current event handler.

### 1.10.3.3 Globals: The Module View

#### 1.10.3.1 Programs in RAP

All code in a report is contained in the Code Module. The Code Module contains all the event handlers for objects contained in the Report. There is also a global section that can contain module level events, declarations, procedures and functions. This part of the Code Module is visible from the Module view of the Code Explorer.

The programs and variables declared in the global section of a module are visible to all subreport code modules in the report. Event handlers appear much as they would in Delphi, with the exception that the Sender parameter is omitted from the signature — this is due to the fact that event handlers can only be assigned to one object in RAP, whereas Delphi allows the same event handler to be assigned to
multiple events.

1.10.3.3.2 Declaring Global Variables

To declare a global variable, right-click on the Code Explorer and select Module. The treeview will change to display the global section.

Click on the Declarations node – this will display two items in the listview: Constants and Variables.

Select the Variables item. If you have already declared some variables, they will be displayed in the Code Editor. If the Code Editor is blank, right-click the Variables item and select New – this will activate the Code Editor and add `var` to the first line.

Declare your variables using standard Object Pascal syntax.

1.10.3.3.3 Declaring Global Constants

To declare a global constant, right-click on the Code Explorer and select Module. The treeview will change to display the global section.

Click on the Declarations node – this will display two items in the listview: Constants and Variables.

Select the Constants item. If you have already declared some constants, they will be displayed in the Code Editor. If the Code Editor is blank, right-click the Constants item and select New – this will activate the Code Editor and add `const` to the first line.

Declare your constants using standard Object Pascal syntax.

1.10.3.3.4 Declaring Global Procedures and Functions

The global section of the Code Module can contain functions visible throughout the module and to any subreports below the current report.

To declare such functions, right-click on the Code Explorer and select Module. The treeview will change to display the global section.

Click on the Programs node – the listview will display any extant functions. If you have not declared any, the listview will be empty.

Right-click on an empty space in the listview – note that the first two menu items are New Function and New Procedure. Selecting either of these items will create a declaration and an implementation stub.

If you select New Function, a new function named `GlobalFunction1` will be added to the listview and the following implementation will be added to the Code Editor:

```pascal
function GlobalFunction1: Variant;
begin
  Result :=
```

© 2004 - 2007 Express Technology Inc.
end;

Likewise, if you select New Procedure, a new procedure named GlobalProcedure1 will be added to the listview and the following implementation will be added to the Code Editor:

```pascal
procedure GlobalProcedure1;
begin
end;
```

### 1.10.3.4 The Code Toolbox

#### 1.10.3.4.1 Overview of the Code Toolbox

The Code Toolbox is a visual code repository. It contains most of the identifiers and code elements that the RAP compiler recognizes.

The Code Toolbox enables you to:

1. **View identifiers** grouped by available **data**, visible **object** properties or **language** features

2. **Generate code by** dragging identifiers into the **Code Editor**. Functions dragged into the editor will generate a place-holder parameter list. For example, the following code is generated when you drag the Copy function into the Code Editor:

   ```pascal
   Copy(S, Index, Count);
   ```

Each tab of the Code Toolbox consists of a treeview and a list of identifiers. The treeview allows you to navigate groups of identifiers listed on each tab.

Identifiers listed in the Toolbox will also display relevant information such as Type, Size, Signature, etc.

Identifiers which appear in the Code Toolbox are registered with RAP via RAP's **extensible architecture**. You can use this architecture to register your own **pass-through functions** and objects with RAP — this allows your new functions to appear in the Code Toolbox and to be recognized by the compiler.
1.10.3.4.2 Data Tab

The Data tab of the Code Toolbox displays data pipelines and fields, allowing you to drag and drop field references into the Code Editor.

Selecting a pipeline from the list will display all the fields in that pipeline as well as data type and size information for the fields.

To insert a field value into the code editing window, select the field and drag it into the Code Editor. The code necessary to retrieve the field value from the pipeline will be generated. For example, dragging the 'City' field from the Code Toolbox pictured above would result in this code:

```plaintext
plClients['City']
```

1.10.3.4.3 Objects Tab

The Objects tab of the Code Toolbox displays report objects and their properties, allowing you to drag and drop properties into the Code Editor.
Selecting an object from the tree will display a list of that object’s properties.

To insert a property into the Code Editor, select the property and drag it into the Code Editor. The code necessary to retrieve the value of the property or call the method will be generated. For example, dragging the ‘AutoSize’ property from the Code Toolbox pictured above would result in the following code:

```csharp
Label1.AutoSize
```

1.10.3.4.4 Language Tab

The Language tab of the Code Toolbox displays RAP language elements, allowing you to drag and drop elements into the Code Editor.
Selecting a category from the tree will display a list of elements for that category.

To insert an element into the Code Editor, select the element and drag it to the Code Editor. The code necessary to reference or use the element will be generated. Note that when you drop an item such as a function call, the function's parameter list is provided. For instance, if you drag Copy into the Code Editor, it will expand as:

\[
\text{Copy}(S, \text{Index}, \text{Count});
\]

When you register pass-through functions with RAP, they will appear automatically in the Function section of the Language Tab.

1.10.3.5 Debugging Options

1.10.3.5.1 CodeSite Support

There is no integrated debugging for RAP – that's a bit beyond the scope of the product. However, we do have an option to provide support for CodeSite – Raize Software Solutions' excellent debugging tool.

In order for you, as a developer, to use this option, you must have a licensed copy of CodeSite installed on your computer.

If you wish to provide this support for your users, your users must have licensed copies of CodeSite installed.

**Important**: RAP’s support of CodeSite in no way changes the licensing agreement of Raize Software Solutions; anyone utilizing RAP’s support of CodeSite must have a properly registered version of CodeSite.

Now, that being said, in order to enable the CodeSite support at run-time, you must add the raCSFuncs unit to your uses clause. This will register the CodeSite pass-through functions with RAP, thus making them available in the Code Toolbox.

To enable the CodeSite support at design-time, you will need to compile and install a package into
Delphi. See the ReadMe.doc file in your RBuilder\Demos\0. RAP\2. CodeSite directory for instructions. Most of the CodeSite calls are made available, some with a few modifications due to RAP’s architecture.

A few notable items:

- CodeSite’s Enabled property is available as the csEnable procedure in RAP. Pass in a boolean value to enable or disable CodeSite.

- Since RAP does not yet support Record types, the SendPoint and SendPointEx functions were changed to accept integers, X and Y instead of a TPoint and the SendRect and SendRectEx functions were changed to accept integers, Left, Top, Right and Bottom instead of a TRect. The pass-through functions will, in turn, map these integers into the proper values when calling the corresponding CodeSite methods.

- There is a demo project showing the use of the CodeSite pass-through functions in RBuilder\Demos\0. RAP\1. CodeSite.

1.10.3.5.1.2 CodeSite Functions

The CodeSite category contains pass-through functions which send messages to the CodeSite viewer. Functions contained within this category are:

- csAddCheckPoint
- csAddSeparator
- csClear
- csEnable
For those unfamiliar with CodeSite, here are some ideas for how to use it with RAP. Note that this is not a primer on using CodeSite; see the CodeSite documentation for that information.

Let us say, for instance, that a TppVariable is not displaying a value you think it should and you want to make sure that a section of code is actually being executed. You might add the following code to the variable's OnCalc event (note that the added code is in red):

```pascal
procedure Variable1OnCalc(var Value: Variant);
var
  lsLabelText: String;
begin
  csEnterMethod('Variable1OnCalc');
  csSendString('TaxRate', plCustomer['TaxRate']);
  if plCustomer['TaxRate'] > 8 then
    begin
      csSendBoolean('Detail.Overflow', Detail.Overflow);
      if Detail.Overflow then
        begin
          lsLabelText := 'Continued...';
```
This is a fairly simple example, but you can see that the liberal use of CodeSite calls can be essentially like stepping through code, looking at values.

There is a demo project showing the use of the CodeSite pass-through functions in RBuilder\Demos\0.RAP\2. CodeSite.

1.10.3.1.4 Conditionally Compiling CodeSite Support

Repeatedly adding and removing CodeSite calls can be troublesome, especially if you have many of them riddling your code. For that reason, the CodeSite calls can be conditionally compiled. All of the CodeSite functions in the raCSFuncs unit have conditional statements around their ExecuteFunction implementations and at the beginning of the unit is the line:

{x$DEFINE CODESITE}

In order to enable the function implementations, remove the "x" from the beginning of the line. To disable them, add the "x". This will allow you to leave all of the CodeSite calls in the code without activating the CodeSite object on your user's machine.

In other words, to enable CodeSite support for your report, do the following:

1. Open the raCSFuncs unit.
2. Scroll to the top of the unit, just below the interface clause.
3. Find the line,

{x$DEFINE CODESITE}  {remove the 'x' to enable CodeSite support}

4. Remove the 'x' at the beginning of the line.
5. Save the unit.
6. Rebuild your project by selecting Build All in Delphi.

To disable CodeSite support for your report while not removing the CodeSite calls themselves, do the following:

1. Open the raCSFuncs unit.
2. Scroll to the top of the unit, just below the interface clause.
3. Find the line,

{$DEFINE CODESITE}  {remove the 'x' to enable CodeSite support}
4. Add an 'x' at the beginning of the line between the '{' and the '$'.
5. Save the unit.
6. Rebuild your project by selecting Build All in Delphi.

1.10.4 Language Reference

1.10.4.1 RAP Language Overview

RAP is a subset of Delphi's Object Pascal. Most language elements in Delphi are recognized by the RAP compiler and can be found in the Code Toolbox. The following is a list of elements contained in the Toolbox.

**Statements:**
- Case statements
- If–then statements
- If–then–else statements
- For loops
- Repeat loops
- While loops

**Data Types:**
- Boolean
- Currency
- Double
- Extended
- Integer
- Single
- Char
- String
- Date
- DateTime
- Time
- Color
- Variant

**Operators:**
- Assignment (':=')
- Boolean (and, not, or, xor)
- Class (as, is)
- Math (-, +, *, /, div, mod)
- Relational (<, <=, <>, =, >, >=)
- String (+)
- Unary (-, +)

**Currently Unsupported Elements:**
- Class declarations
- Arrays
- Record types
- Set types

1.10.4.2 Standard Routines

1.10.4.2.1 String Functions

The String category contains pass-through functions which are string handling routines. Functions contained within this category are:
1.10.4.2.2 Conversion Functions

The Conversion category contains pass-through functions which facilitate type conversions. Functions contained within this category are:

- CurrToStr
- DateTimeToStr
- DateToStr
- FloatToStr
- IntToStr
- StrToCurr
- StrToDate
- StrToDateTime
- StrToFloat
- StrToIntDef
- StrToInt
- StrToTime
- TimeToStr

1.10.4.2.3 Format Functions

The Format category contains pass-through functions which involve string formatting. Functions contained within this category are:

- FormatCurr
- FormatDateTime
- FormatFloat
- FormatMaskText
- FormatString

1.10.4.2.4 DateTime Functions

The DateTime category contains pass-through functions which pertain to date and time handling. Functions contained within this category are:

- CurrentDate
- CurrentDateTime
- CurrentTime
- DayOfWeek
- DecodeDate
- DecodeTime
1.10.4.2.5 Math Functions

The Math category contains pass-through functions which are math routines. Functions contained within this category are:

ArcTan
Cos
Cosh
Cotan
Exp
Frac
Int
IntPower
Ln
Power
Round
Sin
Sqr
Sqrt
Tan
Tanh
Trunc

1.10.4.2.6 Utility Functions

The Utility category contains pass-through functions which are system functions. Functions contained within this category are:

MessageBeep
ShowMessage

1.10.4.3 Classes and Objects

1.10.4.3.1 TraSystemFunction

_Include_ raFunc

**Declaration**

TraSystemFunction = class(TraProgramCallValue)

**Description**

TraSystemFunction is the base class for RAP pass-through functions.

RAP uses TraSystemFunction descendants to define what functions are available to the user. When a TraSystemFunction descendant is registered with RAP using raRegisterFunction, it becomes compilable and appears in the Code Toolbox.

Direct descendants of TraSystemFunction include TraConversionFunction, TraDateTimeFunction, TraFormatFunction, TraMathFunction, TraStringFunction, TraUtilityFunction and TraCodeSiteFunction. These subclasses define the category under which their descendants will
appear in the Code Toolbox.

The descendants of TraSystemFunction override the abstract class function Category to create the categories that appear in the Code Toolbox. To add a function to one of these existing categories simply descend a subclass and descendants will, when properly registered, appear in the Code Toolbox in the appropriate category.

You can also add functions to your own custom categories by deriving a new subclass from TraSystemFunction and overriding Category. Descending new classes from this subclass will cause the functions to be added to your own custom category.

1.10.4.3.2 TraRTTI

Unit
ppRTTI

Declaration
TraRTTI = class(TPersistent)

Description
TraRTTI is the base class for all RTTI support classes used to expose objects to the RAP compiler.

RAP uses TraRTTI descendants to define what objects and which of their members are available to the user and visible to the RAP compiler. When a TraRTTI descendant is registered with the RAP compiler using raRegisterRTTI, new classes, properties and methods become compilable and appear in the Code Toolbox.

Declare a TraRTTI descendant for any object you wish to expose to RAP. In the RCL, the RTTI class hierarchy mirrors the hierarchy of the classes which are made visible to RAP:

TraRTTI
    TObject
        ...
    TComponent
        TraTComponentRTTI
    TppCommunicator
        TraTppCommunicatorRTTI
    TppRelative
        TraTppRelativeRTTI
    TppCacheable
        TraTppCacheableRTTI
    TppBand
        TraTppBandRTTI

1.10.4.3.3 TraParamList

Unit
ppRTTI

Declaration
TraParamList = class(TStringList)

Description
TraParamList is used by the RAP compiler to manage the parameter lists of pass-through functions.

The primary places you will interact with TraParamList are TraSystemFunction, ExecuteFunction,
TraRTTI.GetParams and TraRTTI.CallMethod.

When overriding TraSystemFunction.ExecuteFunction, you will call GetParamValue and SetParamValue in order to read parameter values provided by RAP and to return parameter values back to RAP.

1.10.5 Scaling RAP to Your Users' Needs

1.10.5.1 Scaling RAP to Your Users' Needs

When building an end user solution, you must decide how much of RAP's functionality to expose to your users. RAP allows a continuum from a calculations dialog to a full development environment. Likewise, you can give your users only the base language features included in RAP, or you can expose new, custom pass-through functions, make RAP aware of your own classes and even allow users to display custom forms you design.

The two main properties used for scaling the RAP environment are TppDesigner.RAPOptions and TppDesigner.RAPInterface.

The RAPInterface property controls how the Calc workspace will be presented to the user — as a dialog, a tab or both.

When RAP is presented as a Calc tab in the Report Designer, RAPOptions allows you to specify how much the user is able to view and edit.

1.10.5.2 Defining Your Users' View

With all of the RAP options available, the Calc workspace is a powerful Pascal development environment. For most end users, this may be more than they need (or want). Judging your users' needs, you should define just how much of the RAP IDE to make available to them. Below is a set of descriptions, ranging from simple to full featured, and settings for how to enable those features for your users.

The Calc Dialog — the simplest view.
In this configuration, your users have the ability to assign values for calculations. This means that for any TppVariable added to a report, the user can, by right-clicking over the variable and selecting the Calculations... menu option, display a simple dialog allowing them to assign the value of the variable. This dialog reduces the OnCalc event of the variable component to a simple, one-line assignment of the Value parameter, effectively hiding the entire concept of events from the user.

Set:
TppDesigner.RAPInterface := [riDialog];

Calculations Notebook — variables only.
In this configuration, your users have the same functionality as in the Calc Dialog, but it is available in the Calc tab. In the Calc dialog, the Code Explorer is not shown since, in that dialog, we deal with only one variable at a time. However, in the Calc tab, the explorer is shown, in Variables view, allowing users to navigate through all the variables in the report and edit their values. The Calculations notebook makes it easier to see the code associated with any variables in the report, but still keeps the concept of events hidden from the user.

Set:
TppDesigner.RAPInterface := [riDialog, riNotebook];
TppDesigner.RAPOptions := [roViewVariables, roEditVariables];

Event Tree — components and their events.
In the Events view, the user has access to all the report objects' events, and can code event handlers for them. The events are displayed within the Calc workspace by the Code Explorer.
Set
TppDesigner.RAPInterface := [riDialog, riNotebook];
TppDesigner.RAPOptions := [roViewEvents, roEditEvents];

**Module View — global events and global programs.**
Finally, to expose all the power of the Calc workspace you can add the global settings to RAPOptions. This allows the user to see and edit everything available in RAP. 'Global' refers to programs, functions, variables and constants which can be declared at the report level, and referenced within any event handlers in the report. When the global settings are placed in RAPOptions, the Code Explorer will display a new option under the View menu: Module. When this view is selected, all accessible globals for the current report are shown.
Set:
TppDesigner.RAPInterface := [riDialog, riNotebook];
TppDesigner.RAPOptions := [roViewEvents, roEditEvents, roViewGlobals, roEditGlobals];

**Hybrid Views — some read, some write.**
If you should want to make some parts of the of the workspace editable, some read only and still others hidden, you can set the appropriate options. In this example, the Variables view will be editable, the Events will be read only while the Module view will remain hidden.
Set:
TppDesigner.RAPInterface := [riDialog, riNotebook];
TppDesigner.RAPOptions := [roViewVariables, roEditVariables, roViewEvents];

**Note:** Globals are most appropriate when objects or variables are needed to control report generation. If you want to supply reusable utility procedures, you can add your own pass-through functions to Delphi Object Pascal routines. This is a higher performance option than creating global programs in the report.

1.10.5.3 **Including the Calculations Dialog**

To expose the Calc dialog in the Report Designer in end user solutions, first add raIDE to your uses clause, then include riDialog in TppDesigner.RAPInterface.

1.10.5.4 **Including the Calc Tab**

To expose the Calc tab in the Report Designer in end user solutions, first add raIDE to your uses clause, then include riNotebook in TppDesigner.RAPInterface.

1.10.6 **Extending RAP**

1.10.6.1 **Pass-through Functions**

At the heart of RAP is a run-time compiler that executes your users' code independent of Delphi's compiler. Therefore RAP's compiler has its own independent set of recognizable tokens. In other words, RAP is a subset of Object Pascal.

The method for making the RAP compiler aware of functions is to declare descendants of TraSystemFunction – one for each function – and to register them with the compiler using raRegisterFunction. Each descendant class overrides ExecuteFunction – this method tells the RAP compiler what to do when a function name is encountered in the users' code, thus passing-through the true Delphi functionality to the end user.

For example, when the RAP compiler comes upon the Copy function, it executes the ExecuteFunction method which, in essence, tells RAP how to assign the function's parameter list, then calls Delphi's Copy function.

For examples of registering new pass-through functions, see the main RAP demo: RBuilder\Demos\0. RAP\1. Main\RAP.dpr – reports #31-33. Also see the Adding Functions Tutorial.
1.10.6.2 Adding Functions to the Code Toolbox

While the Language tab of the Code Toolbox contains many useful functions for you and your user, it is sometimes necessary to add functionality.

You can register a new pass-through function with RAP quite easily. The steps are:

1. Identify a Delphi function to call. This can be a function you have written, a DLL call, a WINAPI call or a standard VCL function.
2. Create a TraSystemFunction descendant.
3. Implement this descendant class by:
   - Returning the signature of your procedure in the GetSignature method.
   - Indicating whether the call HasParams and IsFunction via these two boolean methods.
   - And making the actual call to the Delphi function or procedure in the ExecuteFunction method.
4. Register your new function with RAP via a call to raRegisterFunction. Place this call in the initialization section of the unit containing the descendant class.
5. Add this unit to your project.

When you run the application, you will see your new function in the language tab of the Code Toolbox. Also, you will be able to call the function from any RAP event handlers or programs you create.

For a detailed tutorial, see Adding New Functions to RAP.
1.10.6.3 Adding Classes and Properties via RTTI

The report components in the RCL are all surfaced in RAP. In addition, you will notice that the compiler knows about TStrings, TStringList and TList. It is possible to make RAP aware of other objects you may wish to expose within the Calc workspace. For instance, you might want your users to be able to refer to TmyUniversalFinancialComponent.

To do so, see Extending the RAP RTTI. This tutorial will show you not only how to expose your new class, but how to make that class's public properties and methods available as well.

1.10.6.4 Supporting Set Types Using RTTI

This code demonstrates how you can use your TraRTTI descendants to provide support for Set type properties without using Sets.

Version 1.0 of RAP does not support Set types. However, it is possible, using RTTI, to supply support for Set type properties. Take, for instance, TFont's Style property which is of type TFontStyles — a set. In order to support this property via RAP, we have added some boolean properties to TFont in the TraTFontRTTI class, a TraRTTI descendant. Since TFontStyles is a set of fsBold, fsItalic, fsUnderline and fsStrikeout, we have added Bold, Italic, Underline and Strikeout to TFont. In this manner, you can set the boolean properties in RAP and the TraFontRTTI class handles transferring these values to and from the Style property in Delphi's TFont. The following code demonstrates this.

Note: If you are unfamiliar with registering classes with RAP via TraRTTI descendants, you should acquaint yourself with the process by completing the Extending the RAP RTTI tutorial.

First we declare a new TraRTTI descendant, TraTFontRTTI, to provide the new properties.

In GetPropList and GetPropRec, we add the new properties.

In GetPropValue, we transfer values from Delphi's TFont.Style property to the boolean properties:

```pascal
class function TraTFontRTTI.GetPropValue(aObject: TObject; const aPropName: String; var aValue): Boolean;
begin
  ...
  else if ppEqual(aPropName, 'Italic') then
    (if fsItalic is in Style then set TFont.Italic to True)
    Boolean(aValue) := (fsItalic in TFont(aObject).Style)
  ...
end;
```

In SetPropValue, we transfer values from our new properties to Delphi's TFont.Style property:

```pascal
class function TraTFontRTTI.SetPropValue(aObject: TObject; const aPropName: String; var aValue): Boolean;
var
  lFontStyles: TFontStyles;
begin
  ...
  lFontStyles := TFont(aObject).Style;
  ...
  else if ppEqual(aPropName, 'Italic') then
    begin
      (if TFont.Italic then add fsItalic to Style...)
      if (Boolean(aValue)) then
        include(lFontStyles, fsItalic)
      else
        exclude(lFontStyles, fsItalic);
```
TFont(aObject).Style := lFontStyles;
end
...
end;

1.10.7 RAP FAQ

1.10.7.1 RAP Frequently Asked Questions

The Calc tab appears just fine when I'm in Delphi, but when I run the app the tab does not appear. How can I make the Calc tab appear in my running application?
The Calc tab appears in Delphi because the RAP design-time package is installed — therefore ReportBuilder is aware of the RAP IDE. If you want your application to display the Calc workspace, you will need to add raIDE to your uses clause. Additionally, make sure that riNotebookTab is in theRAPInterface property of the Designer. See Defining Your Users' View for more information.

How can I make the Calculations... menu option appear for variable components in my application?
Add raIDE to your uses clause, and make sure that the RAPInterface property of the Designer includes riDialog. See Defining Your Users' View for more information.

What does this message mean: "A Delphi installation has been detected on this machine, but no installation of ReportBuilder Enterprise can be found..."?
This is a licensing issue. Each developer who uses RAP must have a valid ReportBuilder Enterprise license. Our definition of a developer is someone who has Delphi installed on the machine. Therefore, if you have Delphi installed, you will also need ReportBuilder Enterprise installed in order to work with RAP.

What if the same event handler is assigned twice? In other words, if I have assigned MyVariable’s OnCalc event in Delphi, then a user assigns the OnCalc event in RAP, what happens?
In Delphi, when you preview a report at design-time, you will see only the code in the RAP event handler. However, at run-time, both events will fire, first the Delphi event, then the RAP event.

What are those little red and green triangles in the Code Explorer's tree view?
Those are Compilation State Indicators. They show you where your code resides, and whether or not it has compiled successfully. See The Events View for more information.

When descending new TraRTTI classes, do I need to create a new class for every ancestor, or just for the new class?
It's optional. For example, if you wish to register TMyComponent with RAP and there are three levels of hierarchy separating TMyComponent from TComponent, it is unnecessary to declare a new TraRTTI descendant for each of the parents of TMyComponent, unless you wish to make RAP aware of some of the public properties or methods in these levels. See Extending the RAP RTTI for more information.

Since RAP doesn't support Set types, will I be able to set TFont.Style in code?
Yes. We have added properties to TFont to allow you to set these values in code. Normally for a Set type property such as TFont.Style, you might use the Include and Exclude procedures to change its value. In RAP, however, we have added the following boolean properties to TFont for this purpose: Bold, Italic, Normal, Underline and Strikeout. To set a font to bold in RAP, you would say myFont.Bold := True; Likewise to remove style specifications from a font, you would say myFont.Normal := True; We also have an article on building support for Set types into RAP via RTTI.
Index

- A -
Activating an Event Handler Stub 73
Adding Fields 8
adding fields to reports 9
Adding Methods to the Code Toolbox 89
Adding Public Properties via RTTI 90
Adjusting Layout 8
Adjusting Query 8
adjusting report layouts 9
Adjusting Reports 8
Advanced component palette 36
Align or Space toolbar 36
altering reports 9

- B -
bands 41
BarCode 48
Bring to Front 52

- C -
Calc Tab 63, 64, 65, 66, 76
Code Editor 64
Code Explorer 63
Code Toolbox 65, 76
Message Window 66
Calling Procedures and Functions 73
checkbox 49
Code Editor 64
Code Explorer 63, 66, 67
Events View 66
Module View 67
Variables View 66
Code Toolbox 65, 76
Data Tab 65, 76
Language Tab 65, 76
Objects Tab 65, 76
CodeSite Functions 80
CodeSite Support 79
Compiling Event Handlers 74
Conditionally Compiling CodeSite Support 82
Configure 52
Context Sensitive Help 69
Conversion Functions 84
CrossTab 47
Crosstab Wizard 35

- D -
Data component palette 37
data pipeline 19
Data Tab 77
Data Tree 37
Data Types 83
data workspace 14
database 17
DataView 18
DateTime Functions 84
DBBarCode 51
DBCalc 51
DBImage 51
DBMemo 51
DBRichText 51
DBText 52
Declaring Global Constants 75
Declaring Global Variables 75
Declaring Local Constants 73
Declaring Local Variables 73
Defining Your Users' View 87
designing reports 9
detail band 44
draw toolbar 38
duplexing Legalese 71

- E -
Edit menu option 52
Edit toolbar 38
Events View 66
exporting report templates 8
Exposing Input Dialogs Tutorial 72

- F -
Filtering reports 11
Filters - Receiving 8
footer band 45
Format Functions 84

© 2004 - 2007 Express Technology Inc.
Format toolbar 40

-G-
group footer band 45
group header band 43

-H-
header band 42
help with report builder 13
How To 8

-I-
Image 49
importing report templates 8
Including the Calc Tab 88
Including the Calculations Dialog 88

-L-
Label 49
Label Template Wizard 34
Language Tab 78
Line 50
lines menu option 52

-M-
MaintainAspectRatio 53
Math Functions 85
Memo 50
Message Window 66
Method Parameters 73
Module View 67
More Help 8

-N-
Nudge toolbar 40

-O-
Objects Tab 77
Operators 83
Overview of Features 61
Overview of RAP 61
Overview of the Code Toolbox 65, 76

-P-
pagination 53
ParentHeight 53
ParentWidth 53
Position 53
Preview Workspace 16
Programs in RAP 74

-Q-
Query Designer 23
Query Wizard 19

-R-
RAP 61
RAP and Set Types 90
RAP Frequently Asked Questions 91
RAP Pass-Through Methods 88
Region 47
report - altering 9
Report Application Pascal Language Overview 83
Report Builder - Support 5
Report Builder - Tech Support 5
Report Designer 14
Report Tree 40
Report Wizard 28
reports - adding fields 9
reports - adjusting layout 9
reports - data 12
reports - designing 9
reports - exporting templates 8
reports - filters 11
reports - importing templates 8
reports - more help 13
reports - search 12
reports - searching 11
reports help 13
ReprintOnOverflow 53
RichText 50
- S -
Scaling RAP to Your Users' Needs  87
searching in reports  12
searching reports  11
Send to Back  54
Shape  50
ShiftRelativeTo  54
ShiftWithParent  54
SQL  19
Standard component palette  41
Standard toolbar  38
static  55
Stretch  54
StretchWithParent  54
String Functions  83
SubReport  48
summary band  46
Support - Report Builder  5
SuppressRepeatedValues  55
System Variable  50

- T -
Tech Support - Report Builder  5
title band  47
TraParamList  86
TraRTTI  86
TraSystemFunction  85

- U -
Using the CodeSite Functions  81
Utility Functions  85

- V -
Variable  50
Variables View  66
Visible  55

- W -
What is RAP?  61