

# DroidDB<sup>®</sup>

Version 3

---

***SYWARE***<sup>®</sup>  
*Inc.*



# Contents

<b>Introduction .....</b>	<b>7</b>
Overview of DroidDB .....	7
DroidDB Editions.....	8
DroidDB Controls .....	9
DroidDB Terminology.....	11
DroidDB Application Architecture.....	12
<b>Installation and Support .....</b>	<b>14</b>
Installing DroidDB .....	14
Technical Support .....	15
About this Documentation .....	16
<b>PART I: QUICK TOURS .....</b>	<b>17</b>
<b>Quick Tour I: Using a DroidDB Application .....</b>	<b>18</b>
Overview of Quick Tour I .....	18
Quick Lesson 1: Running a DroidDB Application on the Android Device .....	19
Quick Lesson 2: Creating Records .....	20
Quick Lesson 3: Searching for Records .....	23
Quick Lesson 4: Exiting a DroidDB Application .....	24
<b>Quick Tour II: Creating a DroidDB Application .....</b>	<b>25</b>
Overview of Quick Tour II .....	25
Quick Lesson 1: Starting DroidDB's Development Environment.....	26
Quick Lesson 2: Creating a Table.....	28
Quick Lesson 3: Selecting a Table for the Form.....	31
Quick Lesson 4: Creating Tabbed Pages .....	33
Quick Lesson 5: Creating an Edit Box and a Label .....	34
Quick Lesson 6: Creating the Remaining Edit Boxes and Labels .....	38
Quick Lesson 7: Arranging Controls on the Form.....	40
Quick Lesson 8: Creating a Checkbox .....	42
Quick Lesson 9: Creating a Calculated Field.....	43
Quick Lesson 10: Creating a Drop-Down List.....	44
Quick Lesson 11: Creating Radio Buttons .....	47
Quick Lesson 12: Creating a Record Counter .....	49
Quick Lesson 13: Creating a Note Box.....	50
Quick Lesson 14: Creating a Grid Control .....	52
Quick Lesson 15: Creating a Title for the Application.....	54
Quick Lesson 16: Specifying an Initial Sort/Search Order.....	55
Quick Lesson 17: Saving Your Application .....	56
Quick Lesson 18: Exiting the DroidDB Development Environment .....	57
Quick Lesson 19: Testing Your Application .....	58
<b>Quick Tour III: Sharing Data with Desktop Database Applications .....</b>	<b>59</b>
Overview of Quick Tour III .....	59
Quick Lesson I: Downloading a Table to Create a New Application .....	60
Quick Lesson 2: Downloading a Table to Create a New Application – Business Edition.....	63

Quick Lesson 3: Looking at the Synchronization Options – Business Edition .....	67
<b>Quick Tour IV: Getting Acquainted with Macros and Events.....</b>	<b>70</b>
Overview of Quick Tour IV .....	70
Quick Lesson 1: Creating the Form .....	71
Quick Lesson 2: Planning the Macro .....	73
Quick Lesson 3: Opening the Macro Builder .....	74
Quick Lesson 4: Writing a Command to Hide a Control .....	75
Quick Lesson 5: Writing a Command to Show a Control .....	76
Quick Lesson 6: Adding Conditional Branching Logic.....	77
Quick Lesson 7: Attaching the Macro to a Control Event.....	79
<b>PART II: CREATING AND MANAGING YOUR OWN DROIDDB APPLICATIONS.....</b>	<b>81</b>
Overview of Part II .....	82
<b>DroidDB Development Environment.....</b>	<b>84</b>
Starting DroidDB's Development Environment and Beginning a New Application.....	84
Changing the Device Screen Size .....	86
File Menu Commands.....	87
Edit Menu Commands .....	88
Form Menu Commands .....	89
Control Menu Commands.....	90
Development Window Toolbar Buttons .....	91
Development Window Shortcut Keys .....	92
Exiting DroidDB's Development Environment .....	93
<b>Getting Started.....</b>	<b>94</b>
Steps in Creating an Application.....	94
Planning the Application .....	95
About Building Multi-Table/Multi-Form Applications .....	96
<b>Creating and Managing Tables.....</b>	<b>97</b>
Creating Tables.....	97
Creating a Table Using DroidDB .....	98
Downloading a Table from an ODBC-Enabled Application on the Desktop PC.....	101
Recreating a Lost Table.....	103
Adding Columns to an Existing Table.....	104
Deleting Tables.....	106
<b>Using Global Variables.....</b>	<b>107</b>
About Using Global Variables to Store Temporary Values .....	107
<b>Creating and Managing Forms .....</b>	<b>109</b>
Beginning a New Form .....	109
Modifying an Existing Form .....	111
Deleting a Form .....	112
<b>Customizing Forms .....</b>	<b>113</b>
Specifying the Form Size.....	113
Specifying Form Background Color .....	114
Adding Custom Colors to the Color Palette .....	115
Adding a Title.....	116
Adding Tabs.....	117

Customizing or Removing the Menu Bar .....	119
<b>Adding Controls to the Form .....</b>	<b>122</b>
About Controls .....	122
Creating Edit Boxes .....	123
Edit Box Default Values .....	125
Edit Box Money Values .....	127
Edit Box Date/Time Formats .....	128
Creating Note Boxes .....	129
Creating Labels .....	131
Creating Checkboxes .....	132
Creating Radio Buttons .....	133
Creating Drop-Down Lists .....	135
Using Values from Another Table as a Drop-Down List .....	137
Creating Dependent Drop-Down Lists .....	139
Multi-Tiered Dependent Drop-Down Lists .....	142
Creating Navigational Drop-Down Lists .....	149
Creating Calculated Fields .....	151
Creating Command Buttons .....	154
Creating Scribble Boxes .....	158
Using an Image Control to Place a Graphic in the Form Design .....	159
Creating an Image Control to Take Pictures with the Camera .....	160
Creating a Jump Button .....	161
Creating a Lookup .....	165
Creating a Grid Control .....	168
<b>Modifying Control Properties .....</b>	<b>173</b>
Changing Control Properties .....	173
Specifying Font Properties for a Control .....	174
Changing Properties for Multiple Controls .....	175
<b>Arranging Controls on the Form .....</b>	<b>176</b>
Turning Off Snap-to-Grid .....	176
Aligning and Sizing Controls on the Form .....	177
Deleting Controls .....	179
<b>Defining Record Sorts and Searches .....</b>	<b>180</b>
Specifying an Initial Sort/Search Ordering .....	180
Indexing Columns .....	181
Creating Pre-defined Filters for Record Display .....	182
<b>More Form Customization .....</b>	<b>185</b>
Creating Text for the "About Box" .....	185
Allowing Users to Insert and/or Delete Records .....	186
Enabling the Auto Recalc Feature .....	187
Locking the Form Design .....	188
Reports (Business Edition Only) .....	189
<b>Managing, Testing, and Distributing Applications .....</b>	<b>190</b>
Saving an Application .....	190
Opening and Modifying an Existing Application .....	191
Converting a Visual CE Application to a DroidDB Application .....	193

Testing an Application .....	194
Archiving and Deleting Applications .....	195
Creating Distribution Files (Business Edition Only) .....	196
<b>PART III: EXPRESSIONS AND FUNCTIONS .....</b>	<b>199</b>
<b>Using Expressions and Functions.....</b>	<b>200</b>
Introducing Expressions .....	200
Using Date/Time Values in Expressions .....	202
Using If-Then-Else in Expressions.....	203
Using Basic Functions .....	204
Using Advanced Functions .....	205
<b>PART IV: MACROS AND EVENTS .....</b>	<b>211</b>
Overview of Part IV .....	212
<b>Macros and Events: An Overview .....</b>	<b>213</b>
Introducing Macros and Events .....	213
A Sample Macro Script .....	216
Steps in Creating a Macro .....	217
<b>Using the Macro Builder .....</b>	<b>218</b>
Starting the Macro Builder .....	218
Creating a New Macro .....	221
Applying Cut, Copy, and Paste to Macro Commands .....	223
Editing an Existing Macro .....	224
Deleting an Existing Macro .....	225
<b>Creating a Way for Users to Run a Macro .....</b>	<b>226</b>
Creating a Run Macro Command Button .....	226
<b>Triggering Macros with Events .....</b>	<b>227</b>
About Events.....	227
Using Control Events to Trigger a Macro .....	228
Using Form Events to Trigger a Macro.....	229
<b>Command Reference .....</b>	<b>230</b>
Commands by Category .....	230
Commands Organized Alphabetically.....	234
<b>PART V: USING DROIDDB APPLICATIONS .....</b>	<b>267</b>
<b>Using DroidDB Applications.....</b>	<b>268</b>
Overview of Part V .....	268
Starting DroidDB Applications .....	269
Record Menu Commands.....	270
Option Menu Commands.....	271
Exiting DroidDB Applications .....	272
Scrolling Through Records .....	273
Searching for Specific Records in the Table .....	274
Using a Grid Control .....	275
Creating New Records.....	276
Editing Records.....	277

Using Scribble Boxes .....	278
Using Image Controls to Take and Store Photos .....	279
Entering Null Values in a Date/Time or Numeric Field .....	280
Saving Records .....	281
Deleting Records.....	282
Exporting a Table to a Text File .....	283
Importing an ASCII File to a Table.....	284
Clear Table.....	285
Using DroidDB with mEnable.....	286
<b>PART VI: SYNCHRONIZING TABLES .....</b>	<b>287</b>
<b>Table Synchronization (Business Edition Only).....</b>	<b>288</b>
Overview of Part VI .....	288
Creating a Synchronization Configuration for DroidDB Tables and ODBC-Enabled Databases (Business Edition Only) .....	289
Custom Synchronization Options (Business Edition Only) .....	292
Using Timestamp-Based Synchronization (Business Edition Only) .....	294
Synchronizing Multiple Handhelds (Business Edition Only) .....	295
Synchronizing Tables On Command Using DroidDB (Business Edition Only) .....	296
Synchronizing Tables Using DDB_SYNC (Business Edition Only) .....	297
Synchronizing Tables Remotely Using mEnable (Business Edition Only) .....	298
Synchronizing Tables on the Android Device with Excel Spreadsheets on the Desktop PC (Business Edition Only) .....	299

Copyright © 2012-14 SYWARE, Inc. All rights reserved.

SYWARE, DroidDB, Visual CE, Report CE, mEnable, and FoneDB are registered trademarks and sqlceEnable and Mobile 360 are trademarks of SYWARE, Inc. Other brands and their products are trademarks or registered trademarks of their respective holders.



# Introduction

## Overview of DroidDB

Welcome to DroidDB, the fastest and easiest way to create custom form/database applications for Android devices. DroidDB applications allow you to use your Android phone or tablet to collect, organize, display, modify, and share data. You can also synchronize data between DroidDB applications and your favorite ODBC-enabled databases on a desktop PC or remote server.

A DroidDB application is a collection consisting of a SQLite database (which contains one or more related tables) and forms that read and write data in those tables.

You create DroidDB applications while working in DroidDB's development environment on your desktop PC. Each DroidDB form is associated with a table in a SQLite database on the Android device. The database can contain one or a number of related tables. Tables can be empty initially, or you can initialize them with data from another table or ODBC-enabled database.

DroidDB provides 13 types of controls for use on your forms: labels, edit boxes, note boxes, checkboxes, drop-down lists, radio buttons, calculated fields, command buttons, scribble boxes, image controls, jump buttons, lookups, and grids controls.

DroidDB's development environment is simple to use—you specify each control you need and define its properties, including its associated table column, whether it is read-only or read/write, the font and color, and so. You can drag and size the controls to create the form's layout, which appears in the development window as it will display on the Android device.

DroidDB supports all devices running Android 2.1 or later.

The screenshot shows a mobile application interface for recording expenses. It features a table with two columns: 'Expense' and 'Limits'. The 'Expense' column includes a date field (Aug 3, 2011), a text field for the descriptor, a numeric field for the amount (0.00), a checked checkbox for 'Have receipt 15% tip', and a dropdown menu for 'Cost Center' with 'Sales' selected. The 'Limits' column includes a numeric field (0.00) and a radio button selection for categories: Travel (selected), Food, Entertainm, and Other.

*DroidDB enables you to create form/database applications for Android phones and tablets*

## DroidDB Editions

DroidDB is available in the following editions:

*For creating DroidDB applications:*

- **DroidDB Database & Forms Builder – Standard:** This edition enables you to build DroidDB applications for use on Android devices.
- **DroidDB Database & Forms Builder – Business:** This edition has all of the capabilities of the Standard Edition PLUS the ability to synchronize DroidDB apps with ODBC-enabled desktop or server databases. It also includes a Report Writer that can generate reports and email or save them to a file.

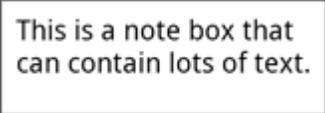
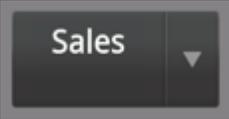
Note: Recipients of redistributed DroidDB apps must obtain their own DroidDB end-user runtime license. As a developer, you must also obtain a runtime license for your Android device; but you can choose either the full end-user or free developer version, as explained below.

*For using DroidDB applications:*

- **DroidDB Runtime – End-User License:** This license enables the DroidDB runtime software on one Android device, making it possible to use DroidDB apps on that device.
- **DroidDB Runtime – Developer License:** This free version of the runtime is useful for creating and testing apps. It is the same as the end-user edition, except that, during use, forms periodically display a pop-up message noting that only the developer version is installed.

## DroidDB Controls

The 13 control types offered by DroidDB are:

Control Type	Description	Example
Label	Text that the user cannot modify. A label can be used to identify other controls, such as edit boxes and note boxes, that do not have their own captions.	
Edit Box	A single-line text box. An edit box can display the value of a table column or global variable, and/or accept the user's entry of a value for a column or variable.	
Note Box	A multi-line text box. A note box can display the value of a table column or global variable, and/or accept the user's entry of a value for a column or variable.	
Checkboxes	A choice or a set of choices from which the user can select none, one, or more than one option.	
Drop-down list	A scrollable list of choices from which the user can choose one option or enter a new option.	
Radio buttons	A set of choices from which the user can select a single option.	
Calculated field	The result of an expression and one or more field values. Operands may be numeric, text, or date/time.	9.60
Command button	A button that the user taps to initiate an action, such as to open another form.	
Scribble box	A box in which the user can draw an image freehand (e.g., a simple sketch or signature).	
Image control	A box that displays images (.bmp., .jpg, or .gif). The user can tap the control to take a photo and store it in the associated table. You can also use an image control to embed a static image, such as a company logo, in the form design.	
Jump button	A button that opens a second form to display rows from a related table.	

## Overview

### Lookup

A value that DroidDB reads from a column in a related table and displays in the current form.

52.03

### Grid Control

Displays rows from a related table and, when a row is touched, opens a second DroidDB form displaying that record. Can also be used to display all records from the current table in a list.

Date	Description
Oct 26, 2011	dinner
Oct 25, 2011	lunch
Oct 24, 2011	airfare

## DroidDB Terminology

A basic DroidDB application consists of a table, form, and optional synchronization configuration.

The **table** is a structure internal to the device that stores the data. Conceptually, it consists of rows and columns. Each *row* is a record that represents one real-world entity (e.g., a customer), and each *column* is a category of data that represents a characteristic of that entity (e.g., customer name). Each cell is a record field that contains the value for that characteristic of that entity (e.g., Acme Widgets). Application users cannot see a table or the data in it, except via a form.

The **form** is the interface that displays data from the table and lets users enter and modify data in the table. It's what users see when they work with your application. The form consists of familiar *controls* that let users add and view data in specific ways – e.g., radio buttons, checkboxes, edit boxes, etc. – as well as calculated fields that compute values. There is generally one control that read/writes to each table column. Your form can also have *macro scripts* that automate tasks and *events* that make the form dynamic. The controls on your form can be laid out in any number of tabbed pages.

The **synchronization configuration** (optional) defines how the changes in the table on the Android device are applied to the corresponding table on the desktop PC or remote server, and vice versa.

### More Complex Applications and Related Tables

An application can have any number of forms and tables. Behind the scenes, an application's tables (whether there are one or many) are stored in the application's database.

An application's tables may be *related*, meaning that a record in one table is related to one or more records in another table by a common value, called the *key*. For example, an Order might be related to many Item records via an order number. DroidDB's relational capabilities make it possible for you to build one form that gathers and displays information from related records, or multiple forms that let users "jump" between detailed views of related records. (More information about DroidDB's relational capabilities and how you can take advantage of them is provided on page 96).

Every application has a primary form. This is the form that opens when the user starts the application on the Android device. (You identify it by giving it the same name as the application.)

While one application can have multiple forms and tables, applications do not share each other's forms, tables, or databases.

## DroidDB Application Architecture

As explained in the previous topic, a DroidDB application consists of table(s), form(s), and optional synchronization configuration.

You define the table, lay out the form, and specify the synchronization settings while working in DroidDB's easy-to-use development environment on the desktop PC. As you work, DroidDB builds the application on the Android device's storage card. For that reason, the desktop PC must be able to "see" the storage card (as removable media with a drive letter or as a "portable device" with no drive letter), and the Android device (or storage card) must be connected to the desktop PC throughout your development session. (Additional information about the desktop PC/Android device connection is provided in the Installation instructions available at <http://www.droiddb.com/install/>.)

As you work in DroidDB's development environment on the desktop PC, you are always working in the context of the current application. If you want to make a change to a different application, you must close DroidDB, restart DroidDB, and open the application that contains the form, table, or synchronization configuration you wish to modify.

### Backing Up Files

As just explained, DroidDB puts all of the application files on the Android device's storage card. It does not save any files on the desktop PC. If you want to backup any of the files, simply copy them from the file system on the storage card to a location on your desktop PC or elsewhere.

### File Structure on the Storage Card

Normally, you do not need to know about the file structure on the storage card, but you may find an understanding of it helpful when troubleshooting, archiving your work, or designing complex applications. Or, you may simply be curious what DroidDB is doing "behind the scenes"...

- Each DroidDB application is contained in a top-level folder on the Android device's storage card. DroidDB automatically creates this folder and assigns it the name you give to the application when you create the application, e.g., `E:\MyApplication` (your drive letter may differ).
- Each application folder automatically contains an SQLite database (.db file) with the same name as the application; e.g., `E:\MyApplication\MyApplication.db`. This database automatically contains all of the tables you create for the application using DroidDB, whether you build them in the development environment or download them from the desktop PC.

In addition, when you create a table, DroidDB automatically stores its initial definition in a system file named `$<tablename>.ddb`. Do not remove the `$*.ddb` files from the application folder or attempt to modify them manually.

- Each application folder automatically contains all of the forms (.ddb files) belonging to the application. The primary form – the one that opens when the user starts the DroidDB application on the Android device—has the same name as the application; e.g., `E:\MyApplication\MyApplication.ddb`. DroidDB automatically assigns this name to the first form you create when you build your application. You can give other forms any name you wish, as long as it is unique within the application and doesn't contain illegal characters.

Each .ddb file contains not only the definition of the form, but also some important information about the table the form is "built on" (the table that the form read/writes data to; the table you select when you build the form). This feature has an important benefit. If a table is lost, you can recreate its structure (but not its data) by simply opening its form in either the DroidDB development environment or on the Android device.

- Each application folder contains an .ini file that stores the DroidDB synchronization settings you specify for the tables in the application database. This file also has the same name as the application; e.g., `MyApplication.ini`.

On the Android device's storage card, there is a top-level folder named `DroidDB` which contains `DroidDB_Info.ini`. This file is created and maintained by the DroidDB runtime system on the Android device. Among other things, the DroidDB development environment looks for this file to detect which removable storage device attached to the desktop PC is in fact the target Android device.

# Installation and Support

## Installing DroidDB

DroidDB requires:

- For DroidDB Database & Forms Builder: A desktop PC or laptop running Windows
- For DroidDB Runtime: An Android device (phone or tablet) running version 2.1 or higher

To install DroidDB:

1. Install the DroidDB runtime system on the Android device.
2. Connect the Android device to the desktop PC, and check the connection.
3. Install the DroidDB Database & Forms Builder on the desktop PC.

For detailed instructions on each of these steps, please go to:  
<http://www.droiddb.com/install/>

## **Technical Support**

Our website, <http://www.droiddb.com>, offers a wealth of information including the latest product development news, tips of the month, and more.

To get help with a specific issue, please submit your request to DroidDB technical support at: <http://www.droiddb.com/contact/forms/request-support.php>.

## **About this Documentation**

This documentation covers both creating and using DroidDB applications.

### **How to Use this Documentation**

Once DroidDB is installed, start with Quick Tour I to see how a sample DroidDB application works on an Android device. Continue with Quick Tour II to create the sample application using the DroidDB development environment on your desktop PC. Take Quick Tour III to learn how to synchronize a DroidDB table with a Microsoft Access database on the desktop PC, and Quick Tour IV to get an introduction to macros and events.

Finally, create and use your own applications using the complete reference information provided for designing and using DroidDB applications. Part II covers creating tables and forms. Part III takes a close look at expressions and functions, which are used in calculated fields and macros. Part IV explains how to use macros and events to make your forms dynamic. Part V covers using DroidDB applications, and, finally, Part VI explains desktop connectivity.

# **Part I: Quick Tours**

# Quick Tour I: Using a DroidDB Application

## Overview of Quick Tour I

In this first quick tour, you will learn how to use a DroidDB application on an Android device using the sample application provided with DroidDB. In just a few minutes use of the sample application, you will become familiar with many of DroidDB's application features.

Quick Tour I includes the following lessons:

- Quick Lesson 1: Running a DroidDB application on the Android device
- Quick Lesson 2: Creating records
- Quick Lesson 3: Searching for records
- Quick Lesson 4: Exiting a DroidDB application

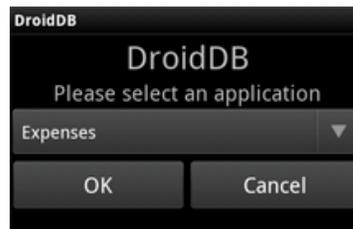
## Quick Lesson 1: Running a DroidDB Application on the Android Device

In this lesson, you will start a DroidDB application and get familiar with DroidDB's application window.

### Note

You must use an Android device to run DroidDB applications.

1. To start the DroidDB runtime software, tap  **DroidDB**.  
DroidDB asks you to select the application you want to use.



2. Choose **Expenses** as shown in the illustration above. Click **OK**.
3. Because the table for this sample application is initially empty, the Expenses application displays an informative message letting you know that you must insert a new record before you can add data. Click **OK** to acknowledge and close the message.

DroidDB displays the EXPENSES application form in its application window.



The DroidDB application window has two menus – **Record** and **Edit**. You open these menus by tapping the device's Menu button.

## Quick Lesson 2: Creating Records

As you create a few records, you will quickly learn the fundamentals of using DroidDB on the Android device.

### Note

Because the table for this sample application is initially empty, all of the form fields are blank. In this case, you must insert a record (as described below) before you can enter data.

On the other hand, if you were opening an application whose table already contained one or more records, DroidDB would automatically display the last record viewed. In that case, you would still need to insert a new record before entering data -- otherwise, you would modify the existing record.

1. Tap the device's Menu button > **Record** > **Insert** to add a new record.

DroidDB creates a new record, and automatically fills in the new record's fields with default values provided by the form designer.

2. In this sample application, the date is automatically today. Just for practice, enter another date:
  - Tap the **Date** field. Droid DB displays a date picker



- Enter a date using the plus and minus buttons or by typing into the fields
- To change back to today's date, tap **Now**
- To confirm your entries, tap **OK**

3. In the **Description** field, enter **Dinner**.
4. Tap the **Amount** field. DroidDB displays a number pad.



- Enter an amount in dollars and cents, such as **49.55**, and tap **OK**.

5. This sample application includes a control that can calculate a 15% tip on the amount. To make the calculation, tap the device's Menu button > **Options** > **Recalc**.
6. Tap the **Have Receipt?** checkbox to indicate that you do have a receipt.

**Tip**

If you need to close the keyboard/keypad so that you can see the whole form, tap the Android device's Back button.

7. Tap the **Entertainment** radio button.
8. In the **Cost Center** drop-down list, leave the default (**Sales**) selected.
9. Tap the **Limits** tab near the top of the screen to go to the next page of the form.
10. In the **Notes** field, enter a brief explanation such as **Client of the Month: Sandy Taylor, SMC Co.**
11. Tap the **Expense** tab to bring the first page of the form back into view.

Now that you have updated the first record, you are ready to save the updates. As you will see, DroidDB automatically saves changes to the current record when you do *any* of the following: insert a new record, move to another record, save the record, close the record, or exit the application by tapping the device's Back button.

12. To insert a new record, tap the device's Menu button > **Record** > **Insert**.

DroidDB saves the changes to the record you just updated, and it creates and displays a new record with default values. Note that the record counter provided by the form designer in the lower left corner of the form indicates that you have created and saved two records.

13. Make entries or accept the default values in each of the controls.

<u>Control Name</u>	<u>Control Type</u>	<u>Entry</u>	<u>Default</u>
Expense Date	Edit box	Accept today's date or tap to enter another date using the date picker	Today's date
Description	Edit box	Enter a short description of the expense	None
Amount	Edit box	Tap to enter an amount	0.00
15% Tip	Calculated field	No user entry – DroidDB calculates the value automatically	15% of the value in the Amount field
Have Receipt?	Checkbox	Accept the default (no receipt) or tap to indicate that you have a receipt	Not checked
Travel/Food/ Entertainment/Other	Radio button	Tap the button that corresponds to the expense category	Travel
Cost Center	Drop-down list	Tap the down-arrow to display the list, then tap an option to choose it	Sales
Notes	Note box	Optional – Enter a multi-line description of the expense	None

14. Save your changes to the record by tapping the device's Menu button > **Record** > **Save**.

15. To view a list of the records you have created so far, tap the **List** tab at the top of the form.



The screenshot shows a mobile application interface for 'Expenses'. At the top, there is a dark header with the title 'Expenses'. Below the header is a tabbed interface with three tabs: 'Expense', 'Limits', and 'List'. The 'List' tab is currently selected and highlighted. Below the tabs is a table with two columns: 'Date' and 'Description'. The table contains two rows of data: 'Aug 3, 2011' with 'Dinner' and 'Aug 23, 2011' with 'Airfare'. Below the table is a large, solid grey rectangular area, likely representing a blurred or redacted section of the application.

Date	Description
Aug 3, 2011	Dinner
Aug 23, 2011	Airfare

16. Tap the **Expense** tab to bring the single-record page back into view.

## Quick Lesson 3: Searching for Records

For the sample Expenses application, the form designer specified a sort/search order based on the values in the Date field. This causes DroidDB to reorder records, based on their dates, as soon as they are saved. (Saving occurs when you insert a new record, move to another record, exit the application, or choose Save.) The record with the most recent date is the first and the record with the oldest date is the last. If no sort/search order had been specified by the designer, records would be organized in a system-defined order.

Because a sort/search order is defined, you can use DroidDB's Search function to go directly to a specific record.

1. Tap the device's Menu button > **Record** > **Search**.

DroidDB displays the date picker.

2. Select the month, day, and year of the record you want, then tap **OK**.

DroidDB displays the first record whose date matches your entry. If there are no records with the date you specified, DroidDB looks backward in time for the next closest date, or if there are none, the last record in the table.

## **Quick Lesson 4: Exiting a DroidDB Application**

You can exit the Expenses application in either of two ways:

- Tap the device's Menu button > **Record** > **Close**. You may have to scroll (swipe) the list of Record options to bring "Close" into view. Or,
- Tap the device's Back button. The first time you touch the Back button, it may close the keypad. If so, tap it again to close the application.

# Quick Tour II: Creating a DroidDB Application

## Overview of Quick Tour II

You can learn the basics of developing DroidDB applications in under ten minutes. In this quick tour, you will create an application that is nearly identical to the sample Expense Tracker program provided by SYWARE, Inc. Creating the application involves the following steps:

- Quick Lesson 1: Starting DroidDB's Development Environment and Beginning Your First Application
- Quick Lesson 2: Creating a Table
- Quick Lesson 3: Selecting a Table
- Quick Lesson 4: Creating Tabbed Pages
- Quick Lesson 5: Creating an Edit Box and a Label
- Quick Lesson 6: Creating the Remaining Edit Boxes and Labels
- Quick Lesson 7: Arranging Controls on the Form
- Quick Lesson 8: Creating a Checkbox
- Quick Lesson 9: Creating a Calculated Field
- Quick Lesson 10: Creating a Drop-Down List
- Quick Lesson 11: Creating Radio Buttons
- Quick Lesson 12: Creating a Record Counter
- Quick Lesson 13: Creating a Note Box
- Quick Lesson 14: Creating a Grid Control
- Quick Lesson 15: Creating a Title for the Application
- Quick Lesson 16: Specifying an Initial Sort/Search Order
- Quick Lesson 17: Saving Your Application
- Quick Lesson 18: Exiting the DroidDB Development Environment
- Quick Lesson 19: Testing Your Application

### Note

To best understand the effect of your development choices, you should use the sample Expense Tracker application before developing a nearly identical application.

## Quick Lesson 1: Starting DroidDB's Development Environment

In this lesson, you'll start the DroidDB Database & Forms Builder (a.k.a., the development environment) on the desktop PC and begin your first app.

### Note

You must have already installed the DroidDB Runtime on the Android device and the DroidDB Database & Forms Builder on the desktop PC.

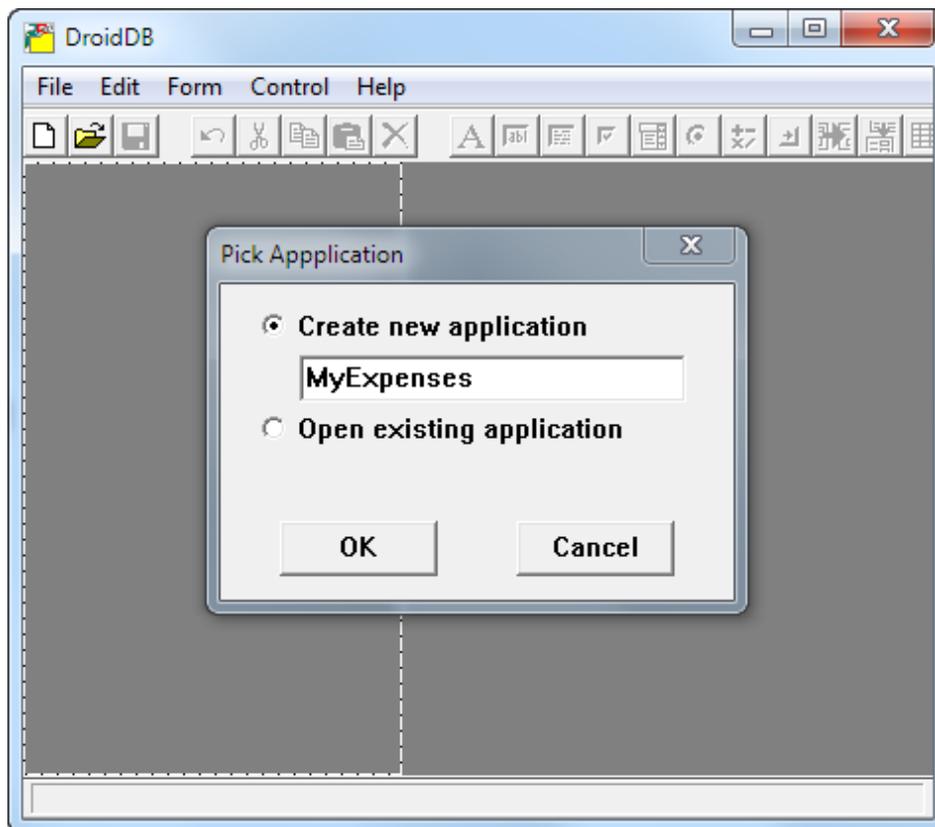
1. Connect your Android device to your desktop PC.

### Note

Whenever you work in the DroidDB development environment, the Android device must be connected to the desktop PC.

2. On the desktop PC, click the Windows **Start** button, then **All Programs > SYWARE DroidDB > DroidDB**. Or, Windows **Start**, then **DroidDB**.

DroidDB asks if you want to open or create an application.

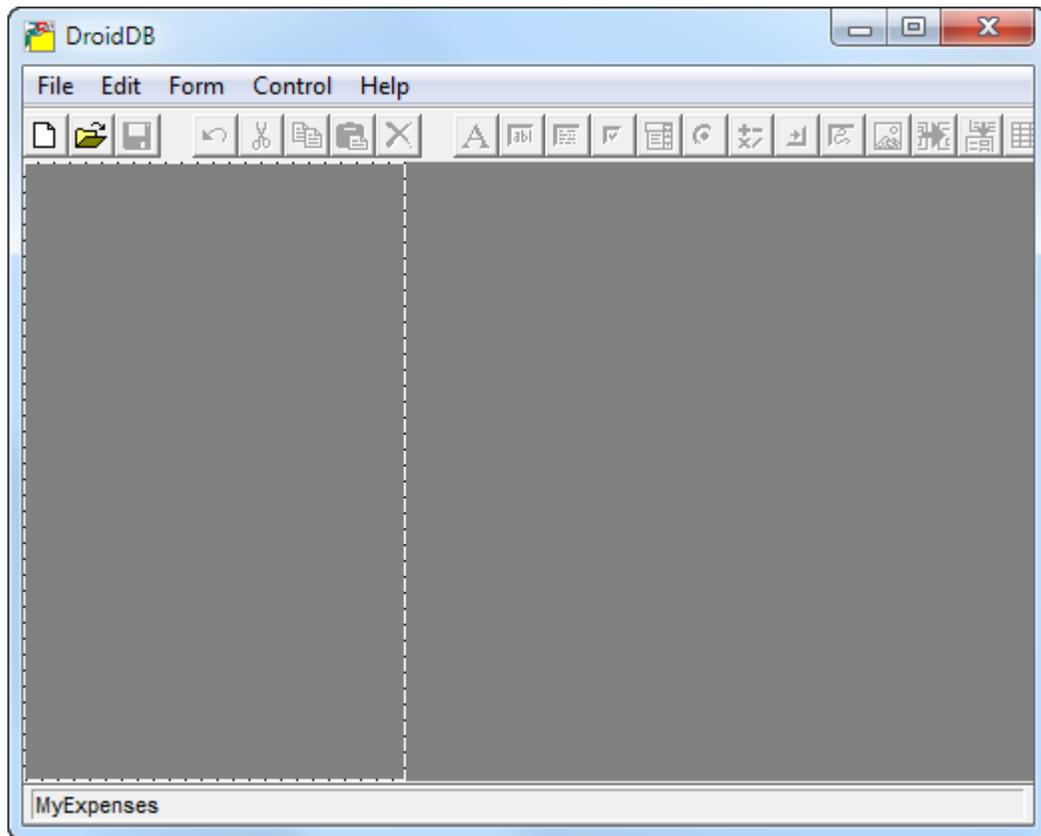


3. Click **Create new application**. Type your new application's name, **MyExpenses**, and click **OK**.

### Note

Once you create an application, all of your work during the development session – creating and selecting tables, building or modifying forms, specifying a synchronization configuration – applies to that application and that application only. If you want to work on a different application, you must exit and restart DroidDB, then pick or create the other application.

DroidDB displays its development environment window.



DroidDB's development environment window has five menus: File, Edit, Form, Control, and Help. You can select menu options by clicking them with the mouse pointer or by pressing the shortcut keys. (To enable the shortcut keys, press the Alt key on your keyboard.)

The toolbar buttons provide a faster way to activate the most frequently used menu options.

The dotted outline in the development window indicates the screen size of the Android device. DroidDB automatically detects the type of Android device connected and displays the default screen size for that type. If you have a device with a larger or smaller display, you can adjust the default using the instructions in the topic "Device Screen Size" on page 86.

DroidDB's default form size matches the screen size. You can make the form height taller than the Android device screen size. (See "Form Size" on page 113 for additional information.)

The status bar in the lower-left corner provides helpful feedback as you build the application. At this point, it displays the name of the application.

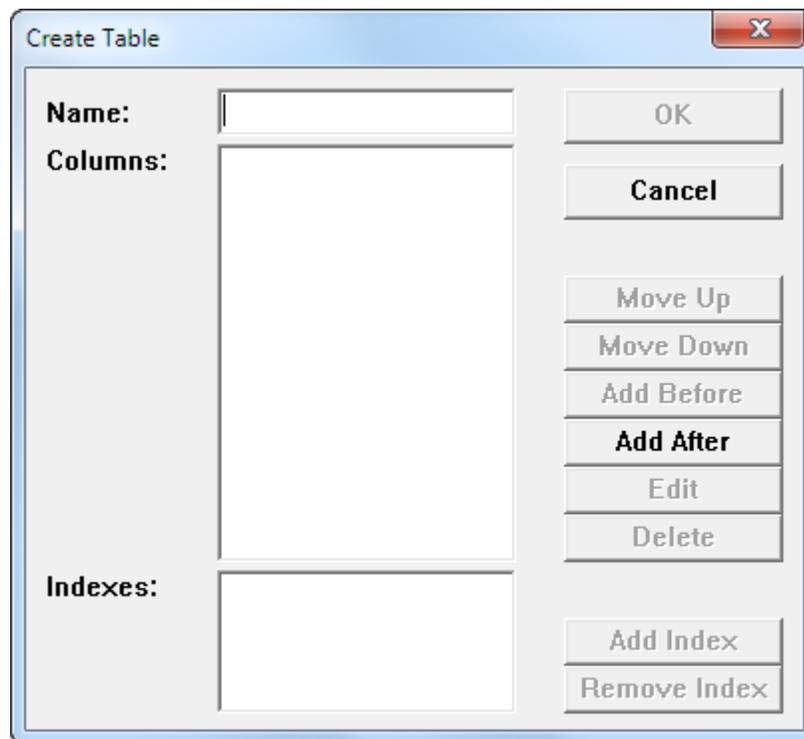
## Quick Lesson 2: Creating a Table

Your next step in creating a new application is to select an existing table or create a new one. For the purposes of this Quick Tour, you will create a new table that is very similar to the one belonging to the Expense Tracker.

You must create one column for each category of data that you want to collect and store in your application.

1. Select **File > Create Table**.

DroidDB displays the Create Table dialog box.



2. For name, type **MyExpensesTable**.

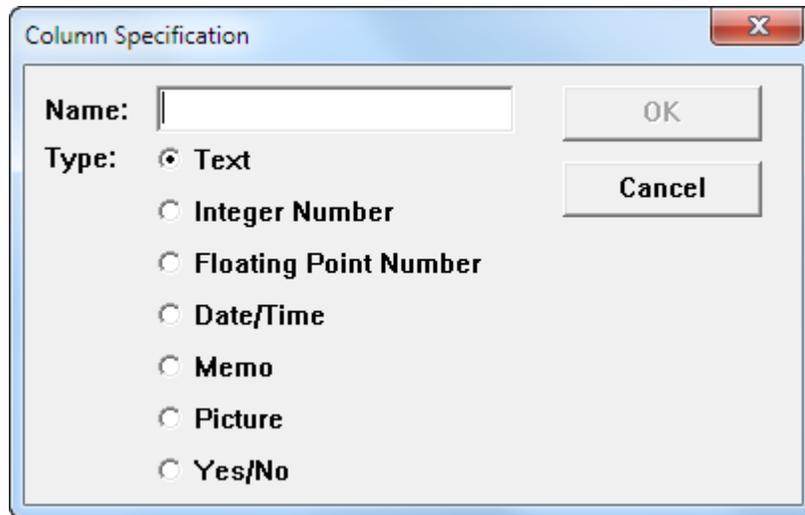
**Note**

The table name and application name may, but need not be, the same.

3. Define the first column:

- Click the **Add After** button

DroidDB displays the Columns Specifications dialog box.

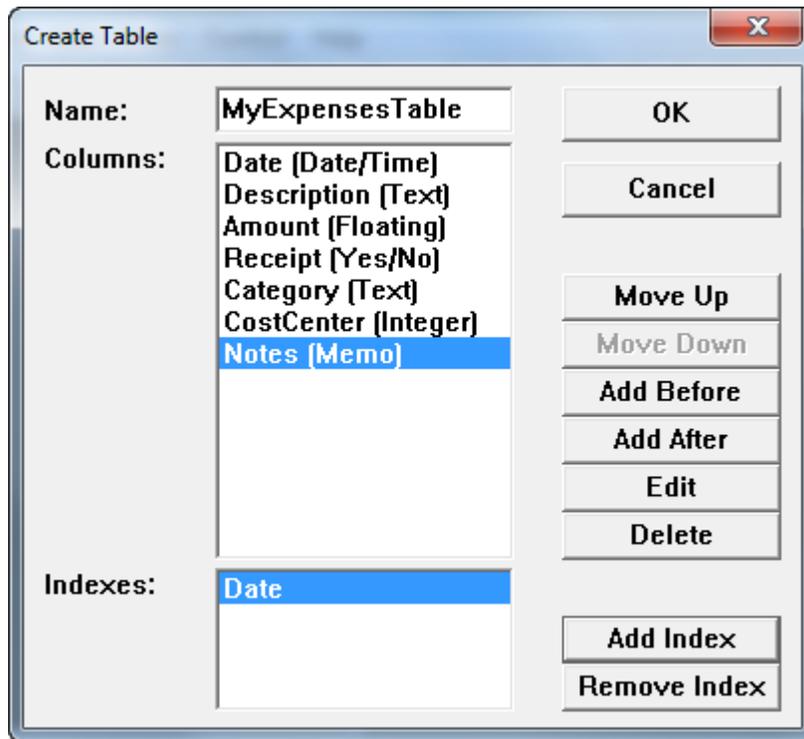


- For name, type **Date**
  - For Type, click **Date/Time**
  - Click **OK**
4. Define the next column:
- Click the **Add After** button
  - For name, type **Description**
  - For Type, click **Text**
  - Click **OK**
5. Define the remaining columns using the same method as described in the previous step:

<u>Control Name</u>	<u>Control Type</u>
Amount	Floating Point Number
Receipt	Yes/No
Category	Text
CostCenter	Integer Number
Notes	Memo

6. "Indexing" a column enables DroidDB to sort the records in the table by values stored in that column:
- Click **Add index**
  - In the dialog that appears, select **Date**
  - Click **OK**

Your table definition so far should look like this:



7. Click **OK** to save your table definition.
8. DroidDB asks if you would like the Form Wizard to create a form for you. (This convenient feature automatically creates edit boxes and labels for each column in the table. You can then modify and rearrange the labels and controls as desired.) However, to get the most practice now, click **No**.

DroidDB's status bar (at the bottom of the development environment window) now displays the name of the application and the name of your new table – indicating that it is selected. As you will see in the following lesson, you build a form on the table that is selected.

You might want to take a break before starting your form design. If so, close DroidDB by selecting **File > Close**.

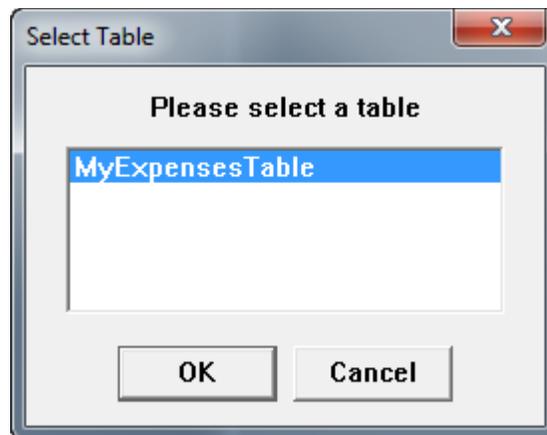
## Quick Lesson 3: Selecting a Table for the Form

In the previous lesson, you created a table. A table is a structure that stores and organizes the data inside the Android device. It is invisible to the user. Users cannot view or modify data stored in a table – except via a form. A form is a program that makes it possible for users to view, add, modify, and delete data in a table. A form is the user’s interface with the data – it’s what they see on the screen.

The first step in building a typical form is to select its table. *If DroidDB is still open from the previous lesson, the table you just created is already selected. In that case, go to Quick Lesson 4.* (You can tell whether or not a table is selected by looking at the development environment’s status bar in the lower left corner – it should display the name of your application and the selected table; in this example, “MyExpenses MyExpensesTable.” Otherwise – if you closed DroidDB after the previous lesson – reopen DroidDB’s development environment on the desktop PC, open the MyExpenses application, and continue below.

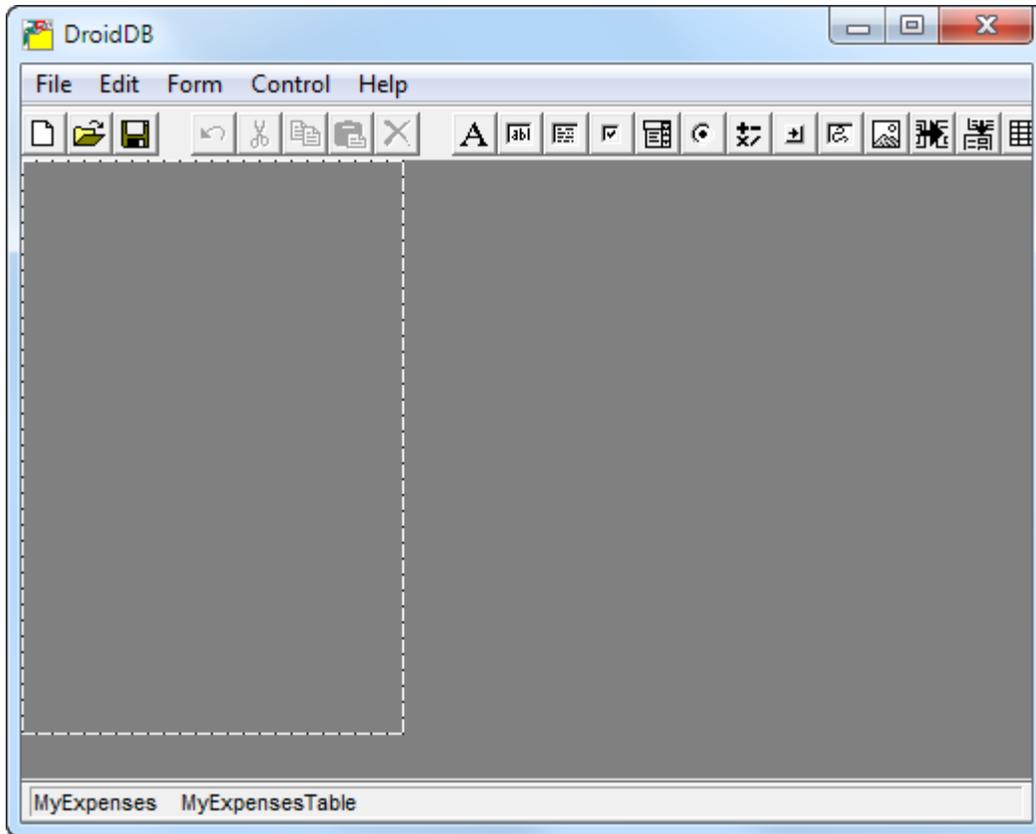
1. From the menu bar, select **File > New Form**.
2. DroidDB asks if the form has an underlying table. Since the MyExpenses application will store records in a table, click **Yes**.

The Select Table dialog box appears.



3. In the Select Table dialog box, click **MyExpensesTable** and **OK**.
4. DroidDB asks if you want the Form Wizard to create the form. For the purposes of this Quick Tour, click **No**.

DroidDB displays the name of the application and the selected table in the status bar in the lower left corner of the development environment window.



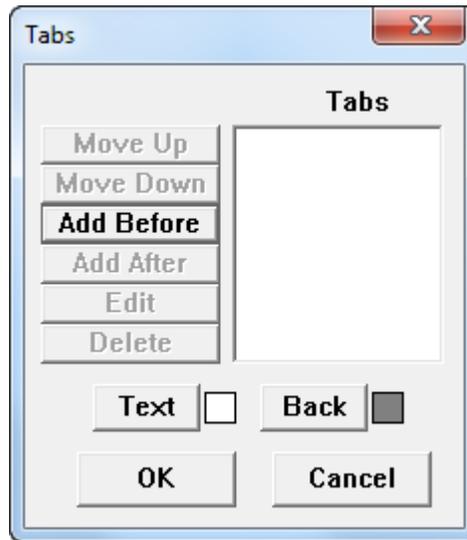
You are now ready to build your first form!

## Quick Lesson 4: Creating Tabbed Pages

You can easily lay out your form as a set of tabbed pages. This is useful if your form is long or complex. Users can quickly navigate different sections by simply touching the tabs, rather than having to scroll.

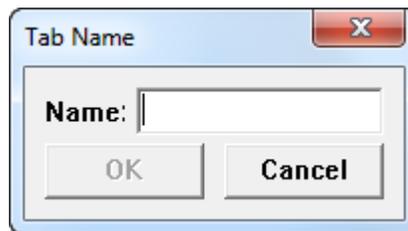
1. Select **Form > Tabs**.

DroidDB displays the Tabs dialog box.



2. Select **Add Before**.

DroidDB displays the Tab Name dialog box.



3. In the Name field, enter the text to appear on the first tab, **Expense**. Then click **OK**.

4. Create the second tab:

- Select **Add After**
- For Name, enter **Limits**
- Click **OK**

5. Create the third tab:

- Select **Add After**
- For Name, enter **List**
- Click **OK**

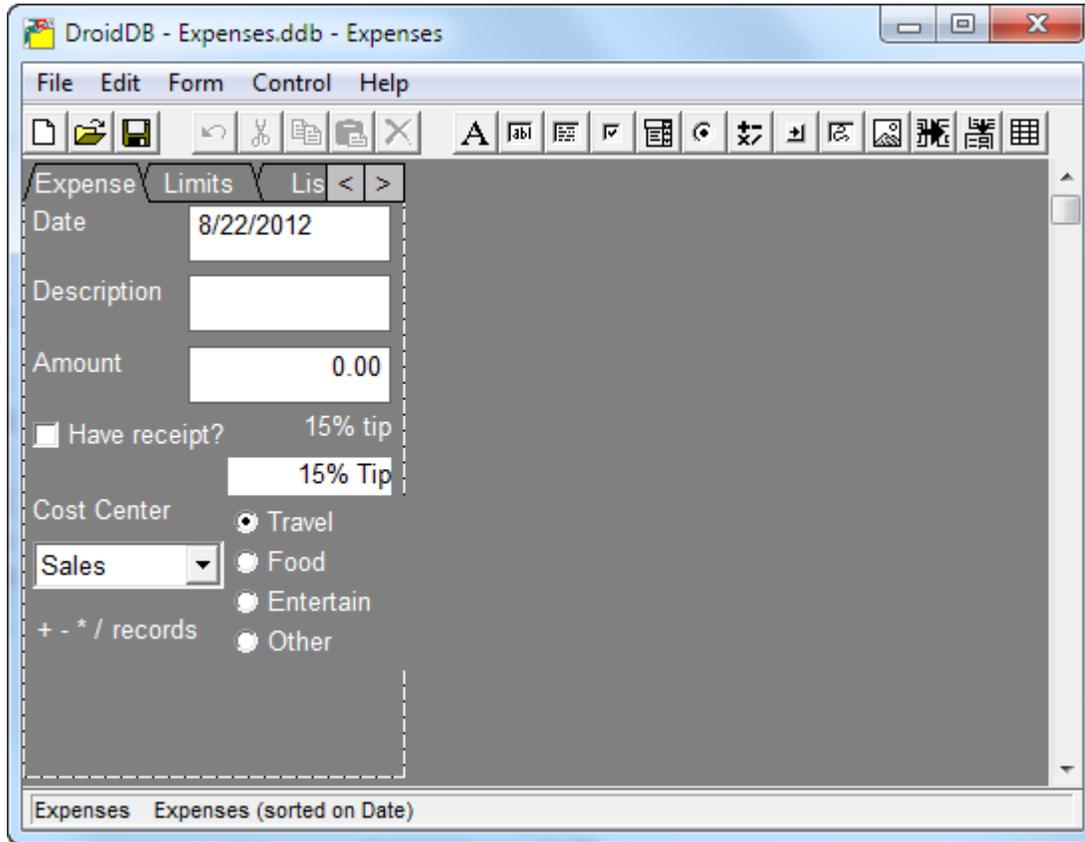
6. Click **OK** to close the Tabs dialog box.

DroidDB places your new tabs at the top of the form in the development environment window. Click on each tab to see how the selected tab appears to move in front of the other.

## Quick Lesson 5: Creating an Edit Box and a Label

You can now begin to create the controls needed on the form.

As you create the application, use the picture below as a reference for the placement of each control in the window.



### Tip

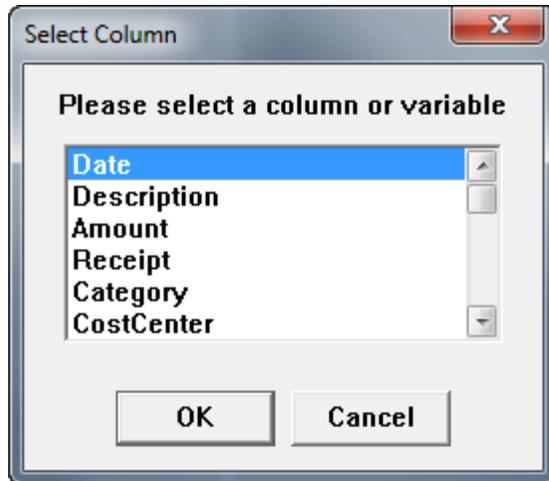
If you make a mistake when creating a control, click the control to select it, then either:

- Edit the control's properties (except the column or control type) by selecting **Edit > Property**, or
- Delete the control by selecting **Edit > Delete**, then recreate the control correctly

1. Click the **Expense** tab to make sure it is selected (it appears to be in front of the other tabs, as shown in the illustration above).

2. Select **Control > Edit** or click the **Edit** button .

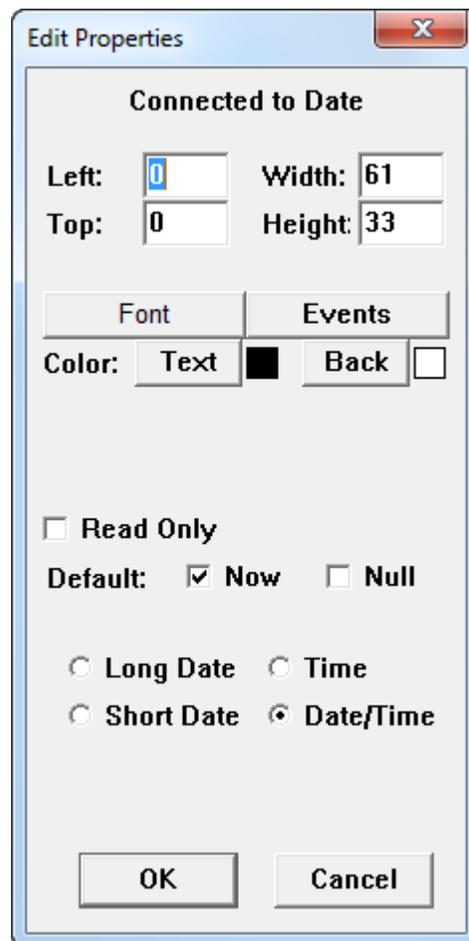
DroidDB displays the Select Column dialog box.



3. Click **Date**, then click **OK**. (Date is the table column to which the edit box relates.)

DroidDB displays the Edit Properties dialog box.

Notice that Droid DB suggests defaults for the edit box's placement (relative to the top-left of the application window) and its dimensions (width and height), as well as the color of the control's font and background. It allows you to define the control as read-only, its default value, and actions to occur when an event takes place, such as the user touches the control.



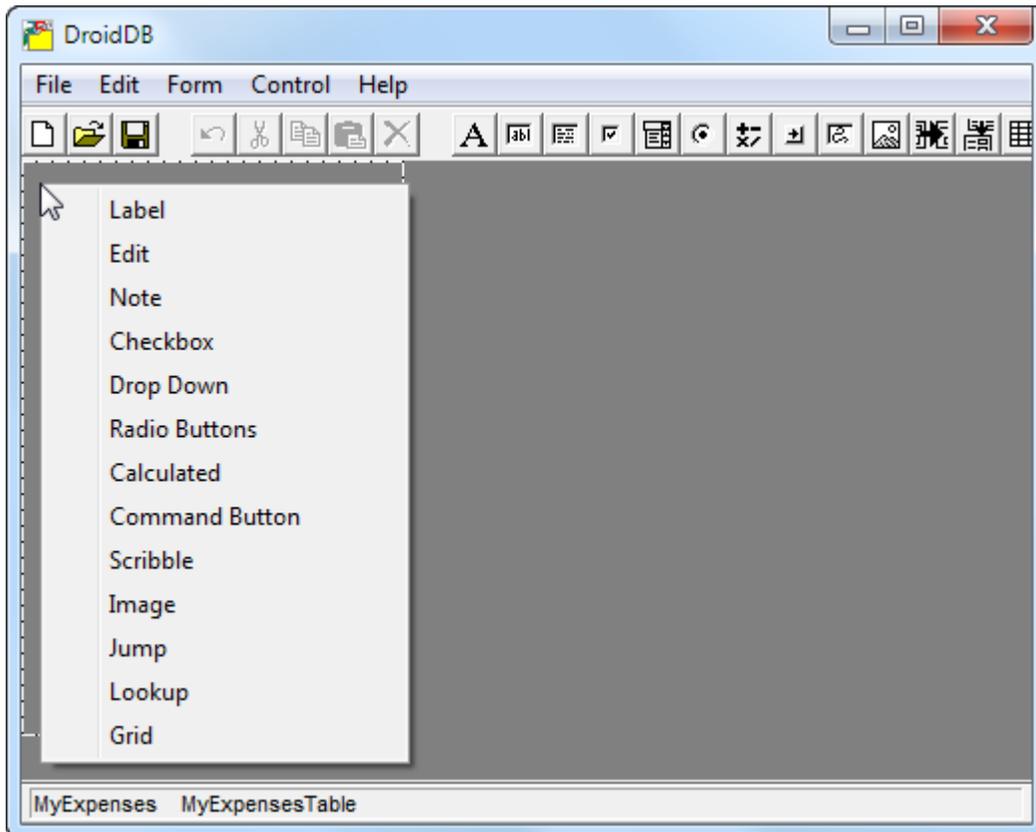
4. Accept all of the faults except the last one: click **Short Date**. Then, click **OK**.
5. Drag the edit box to the approximate location on the form. Note that the box moves in 10 pixel increments. This “snap-to-grid” feature is designed to help you align the controls. If you want to position the box more freely, press and hold the Shift key while you drag the box.

Notice also that the status bar at the bottom of the development environment window displays the control's location and size in pixels. The first pair of numbers (e.g., 0,0) are the x and y coordinates of the control's upper left corner relative to the form's upper left corner. The second pair of numbers (e.g., 61x33) are the control's width and height. The values update as you change the control's location and size.

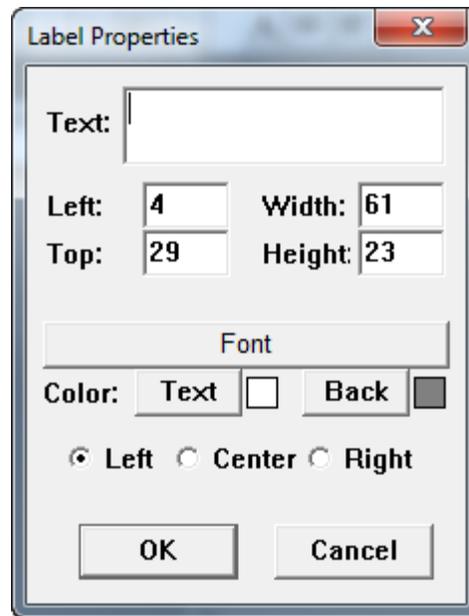
6. Adjust the size of the edit box:
  - Position the pointer on one of the control's “handles” (the small, black squares) so that the pointer turns into a double-headed arrow (if the handles aren't visible, click the control to activate it)



- Click and drag the handle until the control is the desired size
7. Now, create a label for the control, this time using an alternate positioning method:
    - Position the mouse pointer where you want to place the left edge of the label
    - Right-click the mouse, and select **Label** from the menu that appears



DroidDB displays the Label Properties dialog box.



8. Enter **Date** in the Text field, then click **OK**.
9. If necessary, resize and drag the Date label until it more or less aligns with the edit box. You don't need to be exact; you'll align the controls more precisely in a later lesson.

You have already learned the basics of creating controls in DroidDB.

## Quick Lesson 6: Creating the Remaining Edit Boxes and Labels

There are two other edit boxes and labels to be created. One edit box is initially blank for text entries, and the other is defined to accept money entries.

1. Create the “Description” edit control:

- Select **Control > Edit**, or click the **Edit** button , or right-click where you want the edit box to appear and select **Edit**
- Click **Description**, then click **OK** in the Select Column dialog box
- Click **OK** to accept the defaults in the Edit Properties dialog box
- If necessary, drag the edit box to the approximate location

2. Now make a label for the “Description” edit control:

- Select **Control > Label**, or click the **Label** button , or right-click where you want the label to appear and select **Label**
- Enter **Description** in the Text field of the Label Properties dialog box, then click **OK**
- If necessary, drag the label to the approximate position, then resize it

3. Create the “Amount” edit control:

- Select **Control > Edit**, or click the **Edit** button , or right-click where you want the edit box to appear and select **Edit**
- Click **Amount**, then click **OK** in the Select Column dialog box
- Click **Money**, then click **OK** in the Edit Properties dialog box. Note that the “money” selection causes numbers to be displayed with two decimal places
- If necessary, drag the edit box to the approximate location

4. Create the label for the “Amount” edit control:

- Select **Control > Label**, or click the **Label** button , or right-click where you want the label to appear and select **Label**
- Enter **Amount** in the Text field of the Label Properties dialog box, then click **OK**
- If necessary, drag the label to the approximate position, then resize it

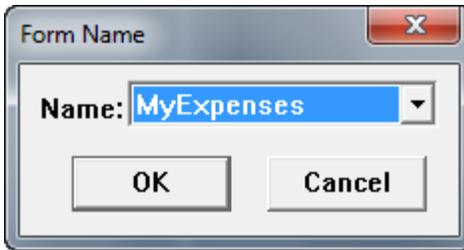
5. Save your work:

- Select **File > Save** or click the **Save** button .

The Form Name dialog box appears with the name of the form filled in.

### Note

You cannot change the name because the first form you create is automatically the primary form, and the primary form always has the same name as the application (in this case, MyExpenses). The primary form is the form that opens when users start your application on the Android device. The sample application has just one form, but more complex applications may have any number of forms.



- Click **OK**.

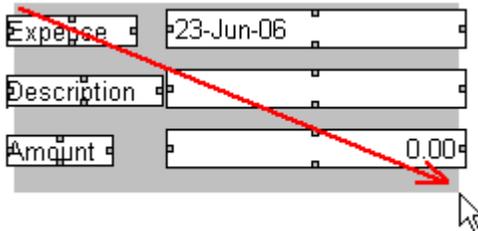
DroidDB saves your new form (MyExpenses.ddb) on the Android device in a folder with the same name as the application, MyExpenses.

You have created all of the edit boxes required on the Expenses form.

## Quick Lesson 7: Arranging Controls on the Form

Now that you've created a few controls, you may want to arrange them more precisely so that your form has a neat appearance. In this lesson, you'll learn about DroidDB's multiple-select and align features, as well as how to specify the exact position of a control in pixels.

1. Align each label with its edit box.
  - Click the date edit box.
  - While pressing the Control key, click on the Date label. Notice that the handles around the first box you selected are solid black, while those around the second box are empty. DroidDB aligns the controls to the first one you select, which is always indicated by the solid boxes.
  - Select **Edit > Align > Bottom**. The bottom edges of the boxes are now aligned.
2. Repeat this process for the remaining labels and edit boxes.
3. Specify a precise horizontal location for the Date edit box:
  - Right-click (or double-click) the date edit box. DroidDB displays the Edit Properties dialog box.
  - In the "Left" field, type **83**. This is the distance in pixels that DroidDB will locate the control from the left edge of the form.
  - Click **OK**.
4. Align the three edit boxes vertically along their left edges:
  - Click the edit box you just positioned, so that DroidDB will use its location as the guide.
  - Select the other two edit boxes (click the edit boxes while pressing the Control key).
  - Select **Edit > Align > Left**.
5. Make all of the edit boxes a uniform width:
  - Click on the date edit box to make it "active."
  - Drag its right handle so that the box is the approximate width shown on the sample.
  - While pressing the Control key, click the other two edit boxes.
  - Select **Edit > Align > Width**.
6. Drag the entire group of controls to a desired position:
  - First, select all of the controls using one of the following methods:
    - ◇ Select **Edit > Select All**, OR
    - ◇ While pressing the Control key, click each of the edit boxes and labels, OR
    - ◇ Left-click the mouse and drag over the controls. The selection area is indicated by a gray box. Release the mouse button. (This method works the same as a Windows marquee select.)



- Then, position the mouse pointer over any one of the selected controls, and press and hold the mouse button.

- Drag the group horizontally and vertically. Notice that the relative position of individual controls stays the same.
- To return the group to their original position (if desired), select **Edit > Undo**.
- To unselect the group, click on some other control.

**Tip**

If you have selected a group of controls, and want to deselect any individual control in the group, press the Control key and click the control.

As you create additional controls in the remainder of this tour, you can apply these techniques to quickly organize the layout of your form.

## Quick Lesson 8: Creating a Checkbox

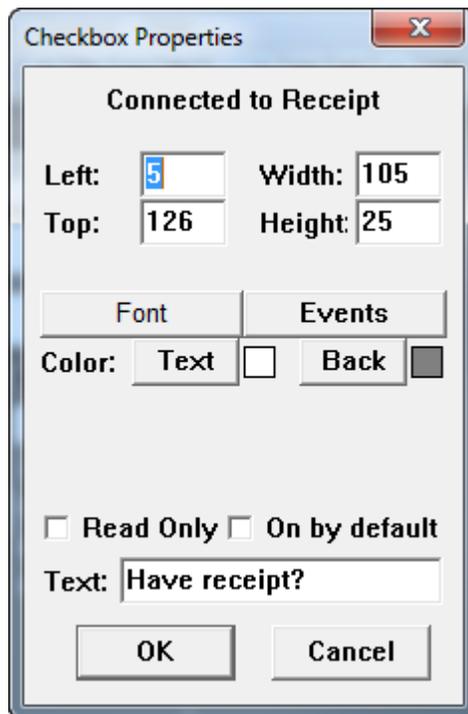
Checkboxes take only seconds to create.

1. Select **Control** > **Checkbox**, or click the **Checkbox** button , or right-click where you want the control to appear and select **Checkbox**.

DroidDB displays the Select Column dialog box.

2. Click **Receipt**, then click **OK**.

DroidDB displays the Checkbox Properties dialog box.



3. Enter **Have receipt?** in the Text field, then click **OK**.
4. If necessary, drag the checkbox to the correct position, then resize it.

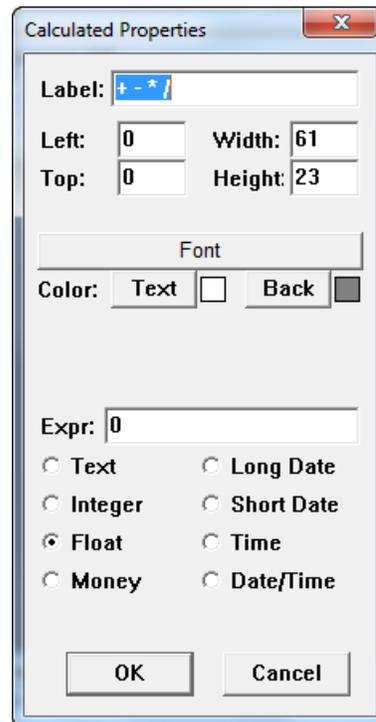
As you created the checkbox, you created the caption (Have Receipt?). You do not need to create labels for checkboxes.

## Quick Lesson 9: Creating a Calculated Field

Calculated fields are great time-savers for anyone who uses your application. They can automatically apply a calculation to a field value and save the results. In this lesson, you will set up a control that automatically calculates a 15% tip on the amount of the expense.

1. Select **Control > Calculated**, or click the **Calculated** button , or right-click where you want the control to appear and select **Calculated**.
2. DroidDB asks if you want the result of the calculation to be saved in a column in the table. For the purposes of this tutorial, click **No**.

DroidDB displays the Calculated Properties dialog box.



3. In the Calculated Properties dialog box:
  - Highlight the default label (+-\*/) and type a new one, **15% tip**. (Your entry here does not appear on the application form. You can think of the Label field as a place to store an optional comment in the development environment.)
  - Enter **Amount \* 15/100** in the Expr field
  - Select the **Money** option to specify the display format for the results, and click **OK**
4. If necessary, drag the field to the correct location.
5. Create a label for the field:
  - Select **Control > Label**, or click the **Label** button , or right-click where you want the control to appear and select **Label**
  - Enter **15% Tip** in the Text field of the Label Properties dialog box, and click **OK**
  - If necessary, drag the label to the correct position, then resize it

## Quick Lesson 10: Creating a Drop-Down List

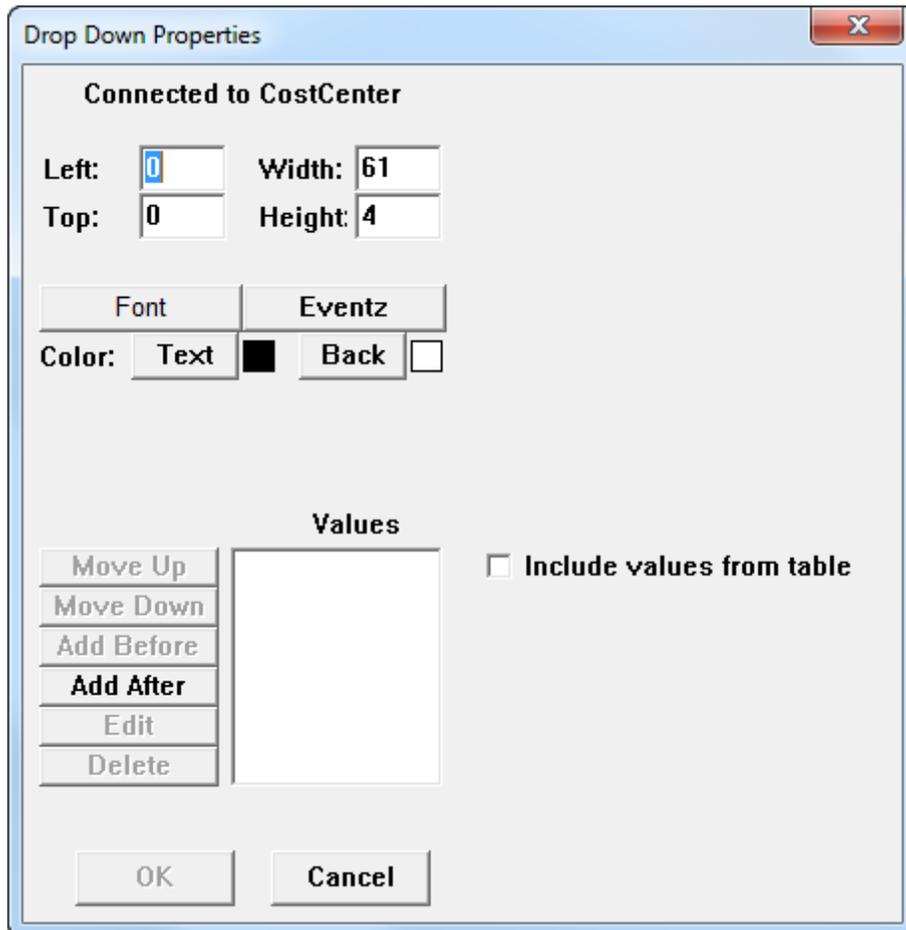
In this lesson, you will create a drop-down list containing six options. One option is defined as the default.

1. Select **Control > Drop Down**, or click the **Drop Down** button , or right-click where you want the control to appear and select **Drop Down**.

DroidDB displays the Select Columns dialog box.

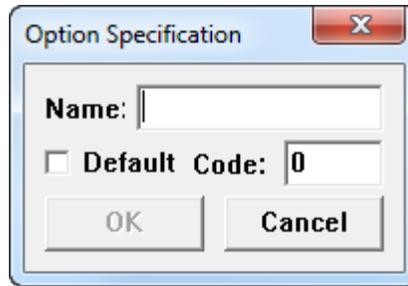
2. Click **CostCenter**, then click **OK**.

DroidDB displays the Drop Down Properties dialog box.



3. Click **Add After** near the bottom left corner of the dialog box.

DroidDB displays the Option Specification dialog box.



4. Create the first value in the drop-down list:
  - Enter **Sales** in the Name field
  - Click **Default** to select the Sales option as the default of the drop-down list
  - Enter code **10045** in the Code field

**Note**

Codes are always/only used when the drop-down is connected to a numeric column in the table. Cost Center is a numeric column. Therefore, you need to provide a numeric code for each drop-down name you enter. The user sees the name (in this case, Sales) on the form, but DroidDB stores the more compact numeric code (in this case 10045) in the table.

- Click **OK**
5. Now create the remaining values in the drop-down list:
    - Click **Sales** to highlight it, then click **Add After**.
    - Enter **Marketing** in the Name field
    - Enter **14578** in the Code field
    - Click **OK**
    - Click **Add After**
    - Enter **Production** in the Name field
    - Enter **14902** in the Code field
    - Click **OK**
    - Click **Add After**
    - Enter **Admin** in the Name field
    - Enter **18012** in the Code field
    - Click **OK**
    - Click **Add After**
    - Enter **R&D** in the Name field
    - Enter **14013** in the Code field
    - Click **OK**
    - Click **Add After**
    - Enter **Other** in the name field
    - Enter **19999** in the Code field
    - Click **OK**
  6. Click **OK** on the Drop Down Properties dialog box.
  7. Drag the drop-down list to the correct location.

8. Now create a label for the drop-down list:

- Select **Control > Label**, or click the **Label** button , or right-click where you want the control to appear and select **Label**.
- Enter **Cost Center** in the Text field of the Label Properties dialog box, and click **OK**
- If necessary, drag the label to the correct position, then resize it

9. To save your work, click the **Save** button .

## Quick Lesson 11: Creating Radio Buttons

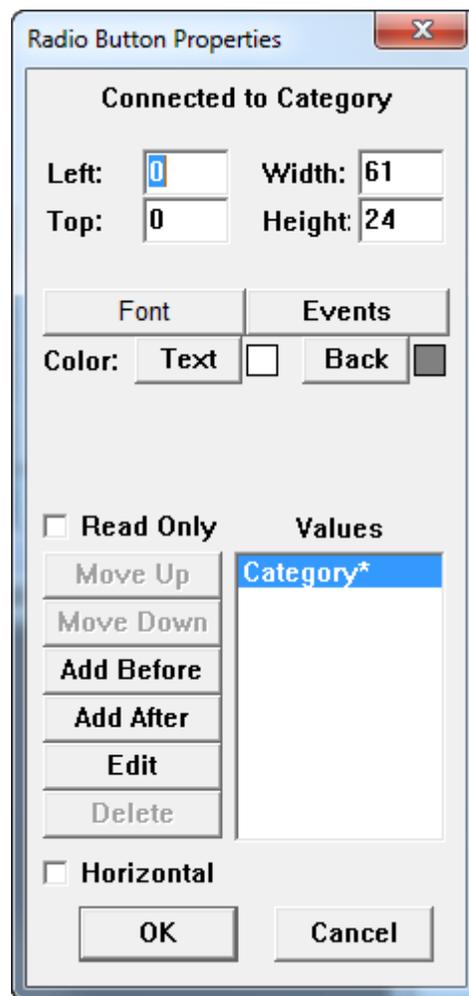
Now you will create a group of four radio buttons, one of which is defined as the default.

1. Select **Control > Radio Buttons**, or click the **Radio Buttons** button , or right-click where you want the control to appear and select **Radio Buttons**.

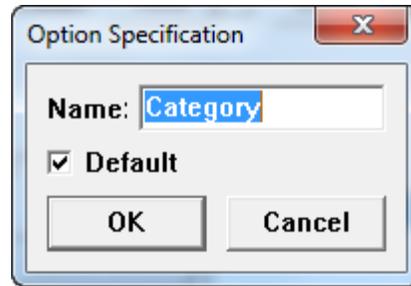
DroidDB displays the Select Columns dialog box.

2. Click **Category**, then click **OK**.

DroidDB displays the Radio Buttons Properties dialog box.



3. In the Radio Buttons Properties dialog box, define the first radio button, which happens to be the default:
  - Click the **Edit** button near the bottom of the dialog box.  
DroidDB displays the Option Specification dialog box.



- In the Name field, replace the word **Category** with **Travel**
- Leave the **Default** box checked, so that **Travel** will be the value for **Category** unless the application user selects a different option
- Click **OK**

DroidDB displays the word **Travel** in the Values column. The asterisk (\*) indicates that it is the default value for the control.

4. Create the three remaining radio buttons:
  - With **Travel** highlighted in the Values column, click the **Add After** button.
  - Enter **Food** in the Name field, and click **OK**
  - Click the **Add After** button
  - Enter **Entertain** in the Name field, and click **OK**
  - Click the **Add After** button
  - Enter **Other** in the Name field, and click **OK**
5. Click **OK** in the Radio Button Properties dialog box to close it.
6. If necessary, drag the control to the correct location, and resize it so all the radio buttons and their names can be clearly read.
7. You may want to save your work again by selecting **File > Save** or clicking the **Save** button .

Because you specified the radio button captions (**Travel**, **Food**, etc.) as you created the buttons, you do not need to create labels for them.

## Quick Lesson 12: Creating a Record Counter

DroidDB comes with many built in functions (predefined calculations) that make it easy to get more meaning out of collected data. One of the simplest and most frequently used functions is `@count`, which automatically tells the number of records stored in the application's table.

1. Select **Control > Calculated**, or click the **Calculated** button  , or right-click where you want the control to appear and select **Calculated**.
2. DroidDB asks if you want the result of the calculation to be saved in a column in the table. For the purposes of this tutorial, click **No**.

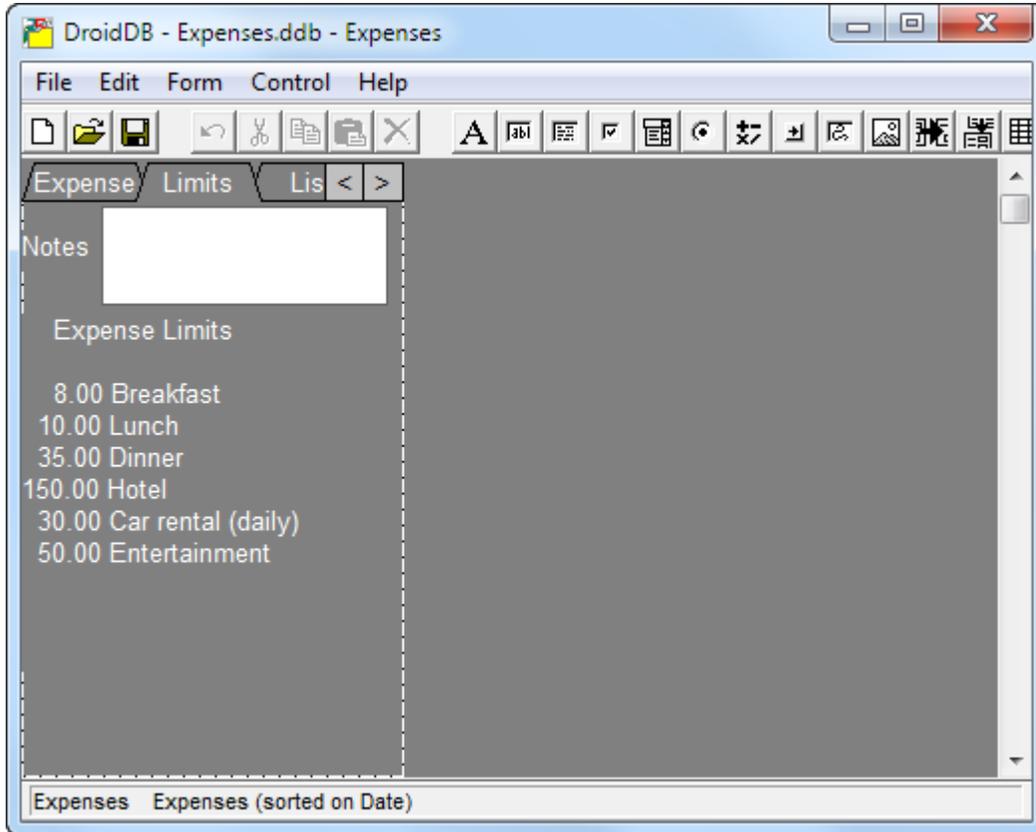
DroidDB displays the Calculated Properties dialog box.

3. In the Calculated Properties dialog box:
  - Enter `@count` in the Expr field
  - Select the **Integer** option to specify the display format for the results
  - Click **OK**
4. If necessary, drag the field to the correct location.
5. Create a label for the field:
  - Select **Control > Label**, or click the **Label** button  , or right-click where you want the control to appear and select **Label**.
  - Enter `records` in the Text field of the Label Properties dialog box, and click **OK**
  - If necessary, drag the label to the correct position, then resize it

You've now created all of the controls on the first tab page.

## Quick Lesson 13: Creating a Note Box

There are just a few items to place on the second tab page. You can use the illustration below as a guide for content and placement.



This page has a note box and extended label.

A note box is really just an edit box that allows multiple lines of text – an edit box allows only a single line of text.

1. Click the **Limits** tab at the top of your form to bring it to the front.

**Tip**

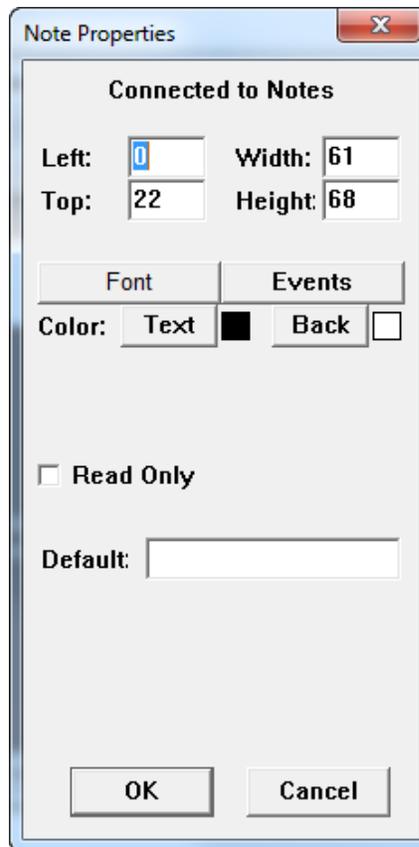
As you create controls, DroidDB automatically puts them on the active tab page. If you change your mind, you can move a control from one tab page to another using cut and paste.

2. Select **Control > Note**, or click the **Note** button , or right-click where you want the control to appear and select **Note**.

DroidDB displays the Select Column dialog box.

3. Click **Notes**, then click **OK**.

DroidDB displays the Note properties dialog box.



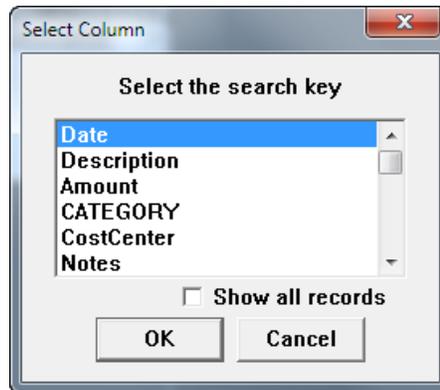
4. Click OK to accept the default values.
5. If necessary, drag the note control to the correct location.
6. Create a label for the note control:
  - Select **Control > Label**, or click the **Label** button , or right-click where you want the control to appear and select **Label**.
  - Enter **Notes** in the Text field of the Label Properties dialog box, and click **OK**
  - If necessary, drag the label to the correct position, then resize it
7. The text that lists the “Expense Limits” is just a long label:
  - Select **Control > Label**, or click the **Label** button , or right-click where you want the control to appear and select **Label**.
  - In the Text field, enter the text shown in the illustration on the previous page. **Tip about alignment:** Each digit is the equivalent in width to two blank spaces. So, for example, to align the decimal points in the sample text, press the Space bar four times before typing “8.00 Breakfast,” two times before “10.00 Lunch,” and no times before “150.00 Hotel”.
  - When you have completed the text, click **OK**.
  - Resize the box so that all of the text is visible, and drag it into position.
8. This is a good time to save your work again. Select **File > Save** or click the **Save** button .

## Quick Lesson 14: Creating a Grid Control

The controls you placed on the Expense and Limits pages let users work with one record at a time. In this lesson, you will create a grid control that displays a handy, scrollable list of all records in the table.

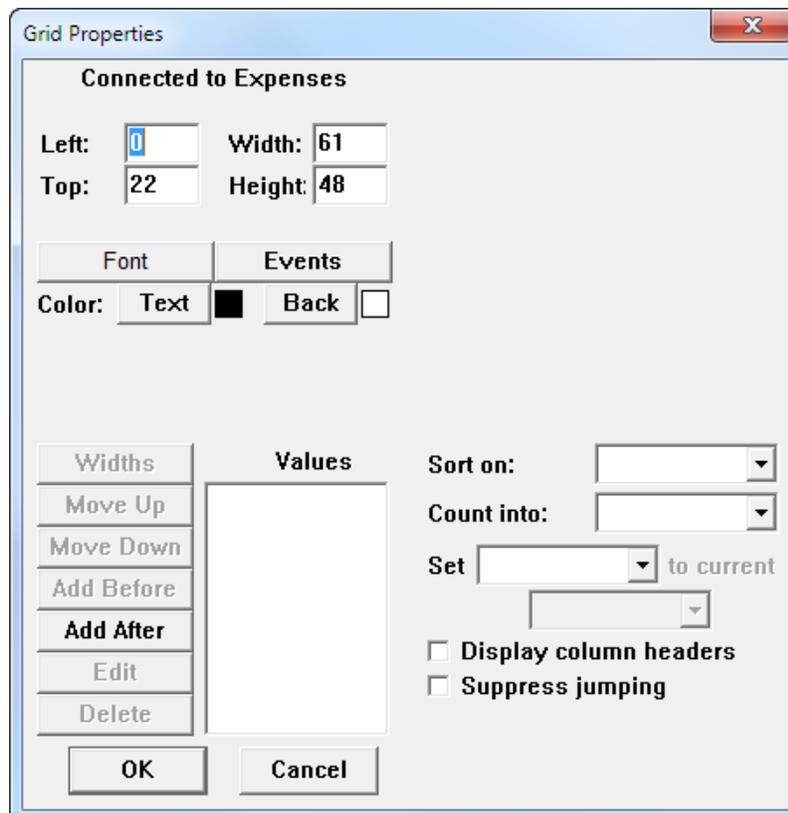
1. Click the **List** tab at the top of your form to bring it to the front.
2. Select **Control-Grid**, or click the **Grid** button , or right-click where you want the control to appear and select **Grid**.

DroidDB displays the Select Column dialog box.

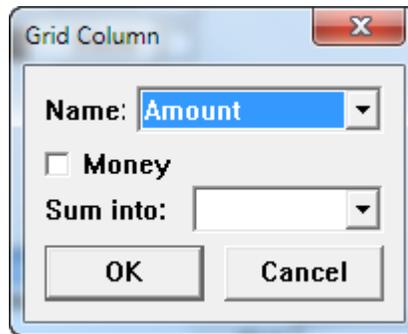


3. Select **Show all records** so that it is checked, and click **OK**.

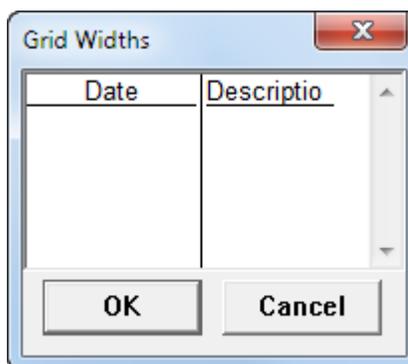
DroidDB displays the Grid Properties dialog box.



4. Specify the first table column to appear in the list display:
  - Click **Add After**.



- Select **Date** in the Name field
  - Click the **Short Date** radio button
  - Click **OK**
5. Specify the second table column to appear in the list display:
    - Click **Add After**
    - Select **Description** in the Name field
    - Click **OK**
  6. In the Grid Properties dialog, select **Display column headers**.
  7. Click **OK** to close the Grid Properties dialog.
  8. DroidDB automatically allocates equal widths to the columns. You probably want to give the Description column a little more space and the Date column a little less:
    - Double-click the grid control you just created to reopen the Grid Properties dialog box
    - Click the **Widths** button.
 DroidDB displays the Grid Widths dialog



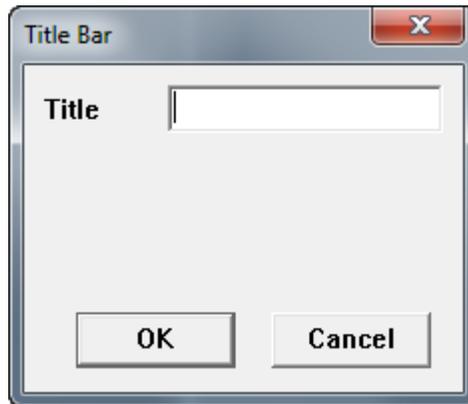
- Point the cursor at the vertical line dividing the two columns and drag it to the left
  - Click **OK**
  - Click **OK** to close the Grid Properties dialog
9. Position and size the control as desired.

## Quick Lesson 15: Creating a Title for the Application

The application's title will display in the title bar in both the development environment window and the application window.

1. Select **Form > Title Bar**.

DroidDB displays the Title Bar dialog box



2. Enter **My Expense Tracker** in the Title field, then click **OK**.

DroidDB displays your title in the title bar.

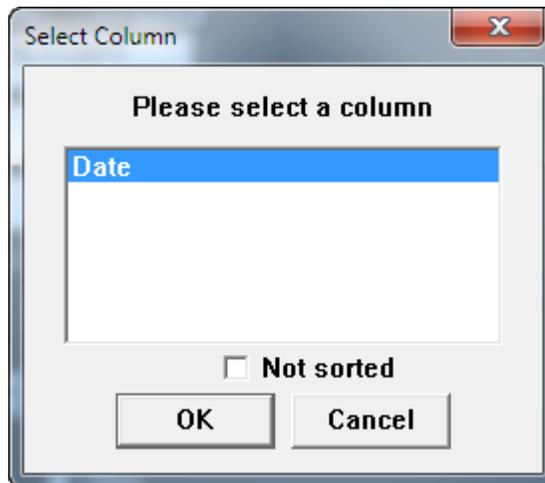
## Quick Lesson 16: Specifying an Initial Sort/Search Order

You have the option to specify an initial sort/search order for record display (the default is unsorted records).

As you saw in the first Quick Tour, the records in the sample application are initially ordered based on the entry in the Date field.

1. Select **Form > Sort On**.

DroidDB displays the Select Column dialog box. Notice that the list includes the only column that you “indexed” in Quick Lesson 2. (Only indexed columns can be used for sorting.)



2. If necessary, click the **Not Sorted** checkbox to remove the checkmark.

3. Click **Date** to select sorting based on the value in the Date column, then click **OK**.

That's all there is to specifying an initial sort/search order.

Congratulations! You have created the Expenses sample application, including most of the controls provided by DroidDB.

## Quick Lesson 17: Saving Your Application

Each time you use the Save command in the DroidDB development environment on the desktop PC, DroidDB saves the application on the Android device.

If you followed the instructions in this Quick Tour, you've already saved the form a number of times.

Select **File > Save** or click the **Save** button .

If you look at the file system on the Android device's storage card, you will see a top level folder with the same name as the application (MyExpenses). Within that folder are the form (MyExpenses.ddb) and the SQLite database (MyExpenses.db).

## Quick Lesson 18: Exiting the DroidDB Development Environment

You can exit the DroidDB development environment when you have finished developing an application, or at any point during application development.

1. Select **File > Close** or click the **Exit** button  on the title bar.

DroidDB asks if you want to save your changes.

2. Click **Yes** to save your changes.

## Quick Lesson 19: Testing Your Application

Test the application on the Android device to be sure that it works the way you intended.

1. Disconnect your Android device from the desktop PC.
2. On the Android device, start the DroidDB runtime by tapping  **DroidDB**.
3. When DroidDB asks you to select an application, choose **MyExpenses** and tap **OK**.  
DroidDB displays your Expense Tracker in the application window.
4. Test your application to be sure that it works correctly. You can use the instructions given in the application-use tutorial (Quick Tour I) to test it.

### Note

Since the table is initially empty, you must insert a new record before you can enter data into the form fields. (The sample application had a pop-up message to this effect. That pop-up message was created using a macro, which we did not include in the tutorial example. Macros are covered in Part IV of this Guide.)

If your application works correctly, you have done a great job! If not, you have probably made a simple mistake or two.

To correct any mistakes, close DroidDB on the Android device, reconnect the device to the desktop PC, and return to the DroidDB development window. If you made a mistake with a control, delete the control and recreate it. If your mistake concerned the title bar or sort/search order, respecify the option correctly. After making your changes, click the **Save** button to save your work. Then repeat the test instructions above.

You have finished Quick Tour II. You are already a DroidDB expert!

### Next:

You can take Quick Tour III which goes step by step through the process of downloading a table from the desktop PC to the Android device and synchronizing the contents of the two tables, as well as Quick Tour IV which introduces macros and events.

Or, you can begin now to create your own applications using DroidDB (page 82).

# Quick Tour III: Sharing Data with Desktop Database Applications

## Overview of Quick Tour III

In this quick tour, you will learn how to create a DroidDB application based on a Microsoft Access table on the desktop PC. The DroidDB table is automatically initialized with the data from the Access table.

If you have the Business edition of DroidDB Database & Forms Builder, you can also set up synchronization between the Access table on the desktop PC and the DroidDB table on the Android device.

### Note

In order to take this Quick Tour, you must have Microsoft Access installed on your desktop PC.

This tour consists of the following lessons:

- Quick Lesson 1: Downloading a Table to Create a New Application
- Quick Lesson 2: Downloading a Table to Create a New Application – Business Edition
- Quick Lesson 3: Looking at the Synchronization Options – Business Edition

## Quick Lesson I: Downloading a Table to Create a New Application

### Note

This lesson applies to Standard Edition users, or to Business Edition users who are not interested in synchronizing the DroidDB table with the Access table. If you are a Business Edition user who is interested in learning about synchronization, skip this lesson and take Quick Lessons 2 and 3 instead.

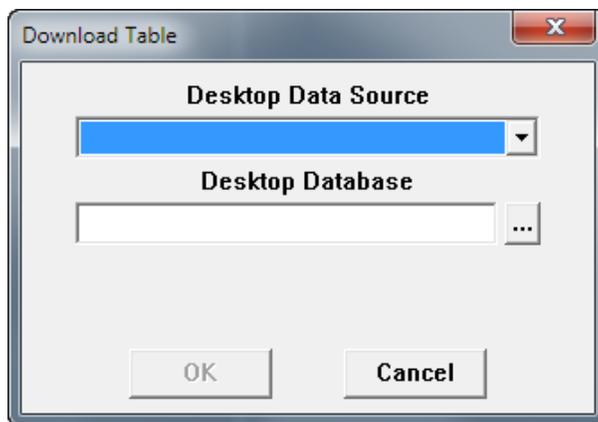
In this lesson, you will download an existing Microsoft Access table and data from the desktop PC to the Android device, and quickly create a DroidDB application for it.

For the purposes of this lesson, SYWARE has provided a sample Access database, "Tutorial.mdb," that contains a table called "Phonebook."

1. Make sure that the Android device is connected to your desktop PC.
2. Start DroidDB's development environment on the desktop PC: click the Windows **Start** button, then **All Programs > SYWARE DroidDB > DroidDB**. Or, Windows **Start**, then **DroidDB**.
3. Create a new application called **Phonebook** (or other name of your choosing).
4. Select **File > Download Table**.

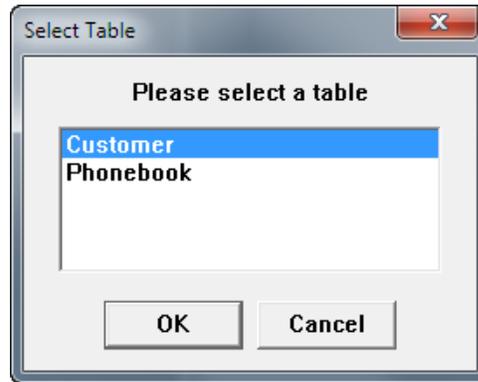
Business Edition users: DroidDB asks if you want to establish synchronization between the tables. For the purpose of this tutorial, click **No**.

DroidDB displays the Download Table dialog box.



5. Establish a connection with the desktop PC database that contains the table you want to download to the Android device:
  - Click on the arrow to the right of the "Desktop Data Source" field, and select **MS Access Database**
  - Click the **Browse** button , and navigate to the folder that contains **Tutorial.mdb**
  - Highlight **Tutorial.mdb** and click **Open**
  - Click **OK** to close the Download Table dialog box

DroidDB displays the Select Table dialog box.

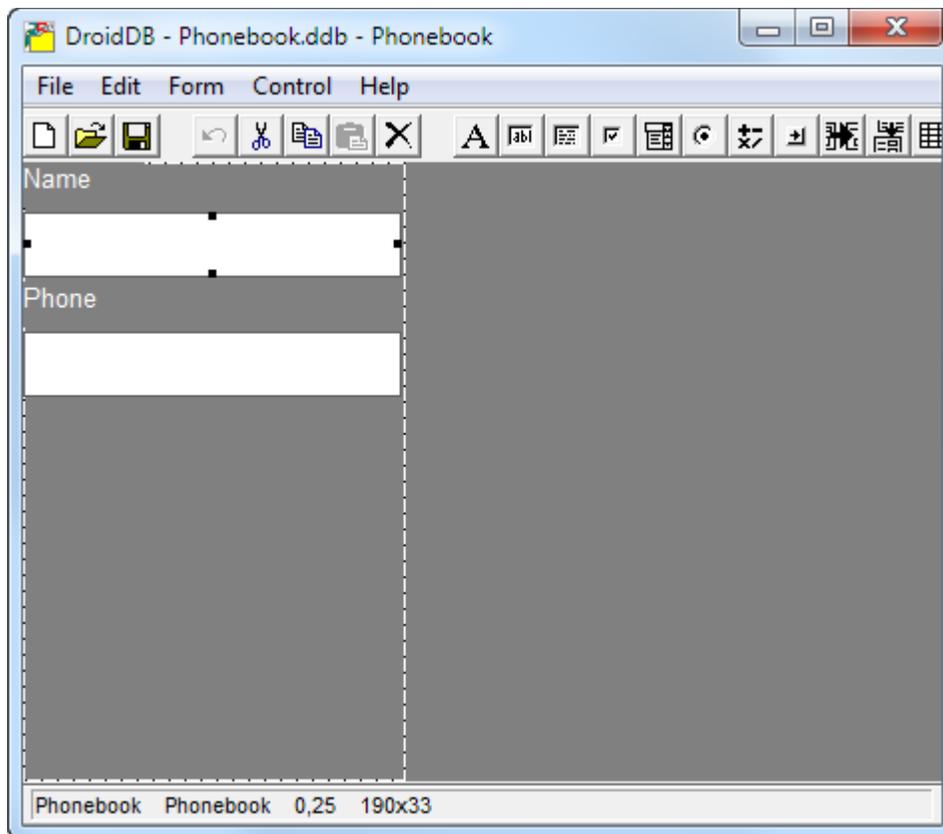


6. Highlight **Phonebook** and click **OK**.

DroidDB downloads the table, and all of the records it contains, from the desktop PC to the Android device.

7. DroidDB asks if you would like the Form Wizard to create a form for you. Click **Yes**.

DroidDB creates controls and labels for you. Adjust the layout as desired using the techniques you learned in the previous quick tour.



8. To save the form, select **File > Save** or click the **Save** button .

DroidDB displays the Form Name dialog box.



9. Click **OK**. (You cannot change the name of the first form you create for an application.)
10. Close the development environment by selecting **File > Close**.

That's all there is to it! You have created a DroidDB application for the Android device that utilizes the same table structure and data as an Access table on your desktop PC.

**Next:**

(For Business Edition users only) Quick Lessons 2 and 3 are similar to the lesson you just completed, but they also include table synchronization.

(For all users) Quick Tour IV takes a closer look at two features for making your forms dynamic and easier to use: macros and events.

## Quick Lesson 2: Downloading a Table to Create a New Application – Business Edition

### Note

This lesson applies to Business Edition users only.

In this lesson, you will download an existing Microsoft Access table and data from the desktop PC to the Android device, and quickly create a DroidDB application for it. You will also set up synchronization for the two tables so that the contents of the table on the desktop PC stay current with those on the Android device, and vice versa.

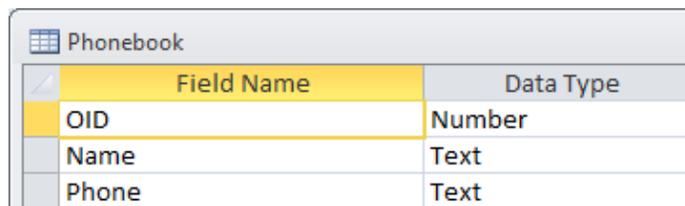
For the purposes of this lesson, SYWARE has provided a sample Access database, "Tutorial.mdb," that contains a table called "Phonebook."

1. Before you can apply synchronization to an Access table on the desktop PC for the first time, you must add an "OID" column to it. (DroidDB automatically adds a corresponding column to the table on the Android device and uses the columns to keep records in the two tables in sync.) To do this:

- Start Microsoft Access on the desktop PC
- Open **Tutorial.mdb** (typically stored in \Program Files\SYWARE DroidDB)
- Select **Phonebook** table and go to the Design view
- Add a new field definition to the *top of the list*, with the following specifications:

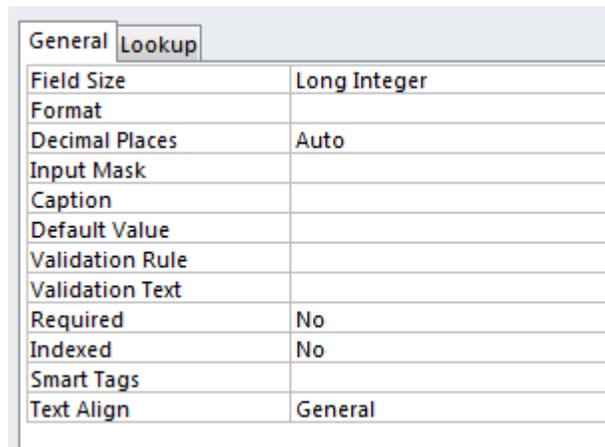
Field Name: **OID**

Data Type: **Number**

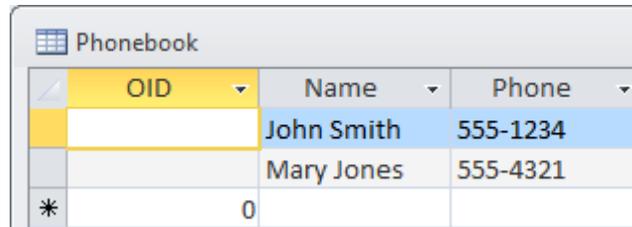


Field Name	Data Type
OID	Number
Name	Text
Phone	Text

- ◇ In the General tab in the Access Design view, turn off indexing. Leave the remaining default values in place.



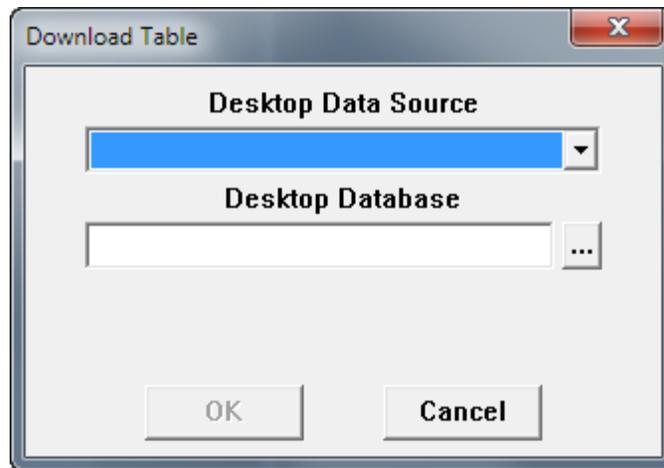
Property	Value
Field Size	Long Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	No
Smart Tags	
Text Align	General



OID	Name	Phone
	John Smith	555-1234
	Mary Jones	555-4321
*	0	

- Save your changes and close Access.
2. Make sure that the Android device is connected to your desktop PC.
  3. Start DroidDB's development environment on the desktop PC: click the Windows **Start** button, then **All Programs > SYWARE DroidDB > DroidDB**. Or, Windows **Start**, then **DroidDB**.
  4. Create a new application called **Phonebook\_sync** (or other name of your choosing).
  5. Select **File > Download Table**.
  6. DroidDB asks if you want to establish synchronization between the tables. For the purposes of this lesson, click **Yes**.

DroidDB displays the Download Table dialog box.



7. Establish a connection with the desktop PC database that contains the table you want to download to the Android device:
  - Click on the arrow to the right of the "Desktop Data Source" field, and select **MS Access Database**
  - Click the **Browse** button , and navigate to the folder that contains your modified version of **Tutorial.mdb**.
  - Highlight **Tutorial.mdb** and click **Open**
  - Click **OK** to close the Download Table dialog box

DroidDB displays the Select Table dialog box.

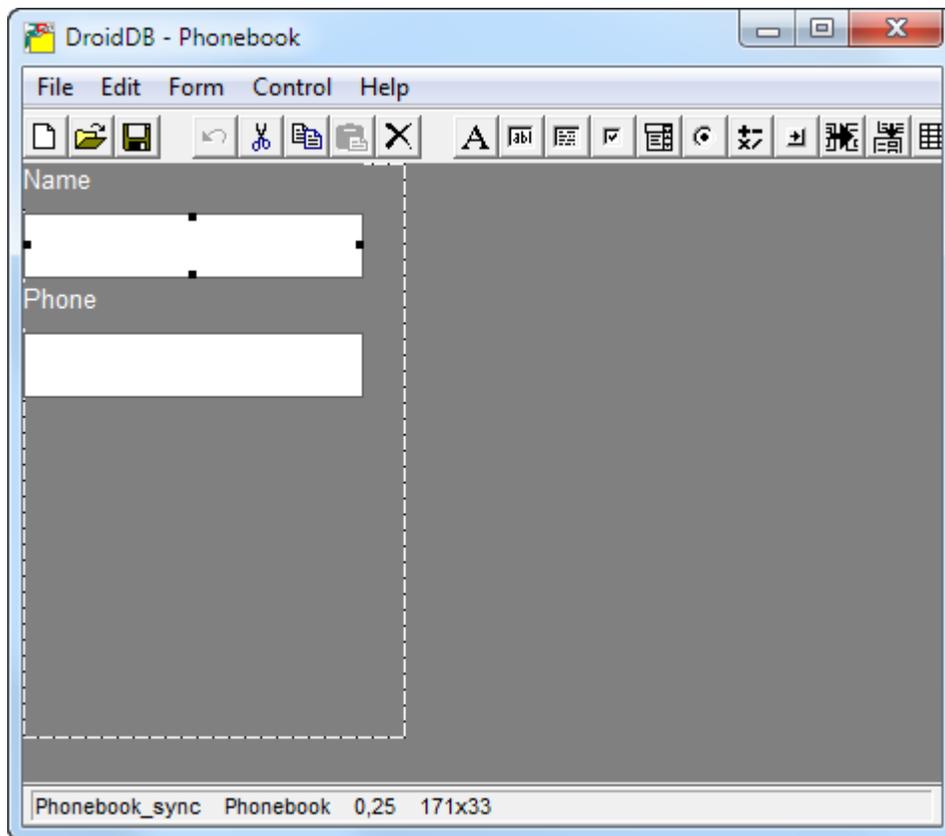


8. Highlight **Phonebook** and click **OK**.

DroidDB downloads the table, and all of the records it contains, from the desktop PC to the Android device.

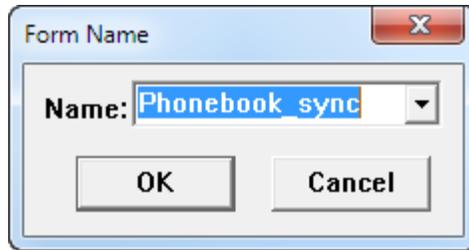
9. DroidDB asks if you would like the Form Wizard to create a form for you. Click **Yes**.

DroidDB creates controls and labels for you. Adjust the layout as desired using the techniques you learned in the previous Quick Tour.



10. To save the form, select **File > Save** or click the **Save** button .

DroidDB displays the Form Name dialog box.



11. Click **OK**. (You cannot change the name of the first form you create for an application.)

12. Close the development environment by selecting **File > Close**.

That's all there is to it! You have created a DroidDB application for the Android device that utilizes the same table structure and data as an Access table on your desktop PC, and you have set up synchronization for the two tables using DroidDB's default synchronization settings.

In the next lesson, you'll explore how the synchronization settings for the tables can be customized.

## Quick Lesson 3: Looking at the Synchronization Options – Business Edition

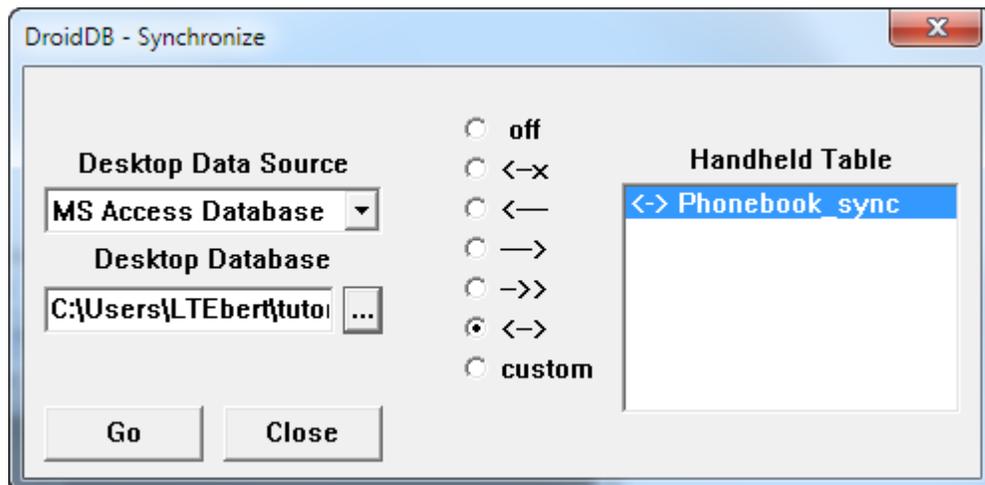
In this lesson, you'll explore the options DroidDB provides for synchronizing tables on the desktop PC and Android device (aka, handheld). These options allow you specify whether updates are applied in just one or both directions, and to fine-tune how DroidDB handles specific differences between records in the two tables.

### Note

Synchronization is available to Business Edition users only.

1. Open DroidDB's development environment on your desktop PC.
2. Open the Phonebook\_sync application that you created in the previous lesson.
3. Select **File > Synchronize**.

The Synchronize dialog box appears. It allows you to view all of the synchronization settings currently in place, to modify the settings, and to launch synchronization.



The fields on the left side of the dialog box pertain to the desktop PC tables; those on the right, to the tables on the Android device.

- **Desktop Data Source** and **Desktop Database** together specify a connection to the ODBC-enabled data source on the desktop PC. All of the desktop PC tables you want to synchronize must be in this one data source, which is typically an Access database (.mdb or .accdb file).

### Note

Once you have completed the Quick Tour, you can specify a different data source for your own desktop PC tables. But you must keep all of the tables you want to synchronize in that one new data source from then on.

- **Handheld Table:** This listbox contains the names of all tables that are eligible for synchronization in the current application folder on the Android device. The symbol to the left of the table name in the listbox indicates the synchronization option currently specified for that table. The symbols have the following meanings:

[blank]	Same as "off." Prevent synchronization of the handheld table with the desktop PC table. Any changes made to records in one table are NOT applied to the other.
---------	--

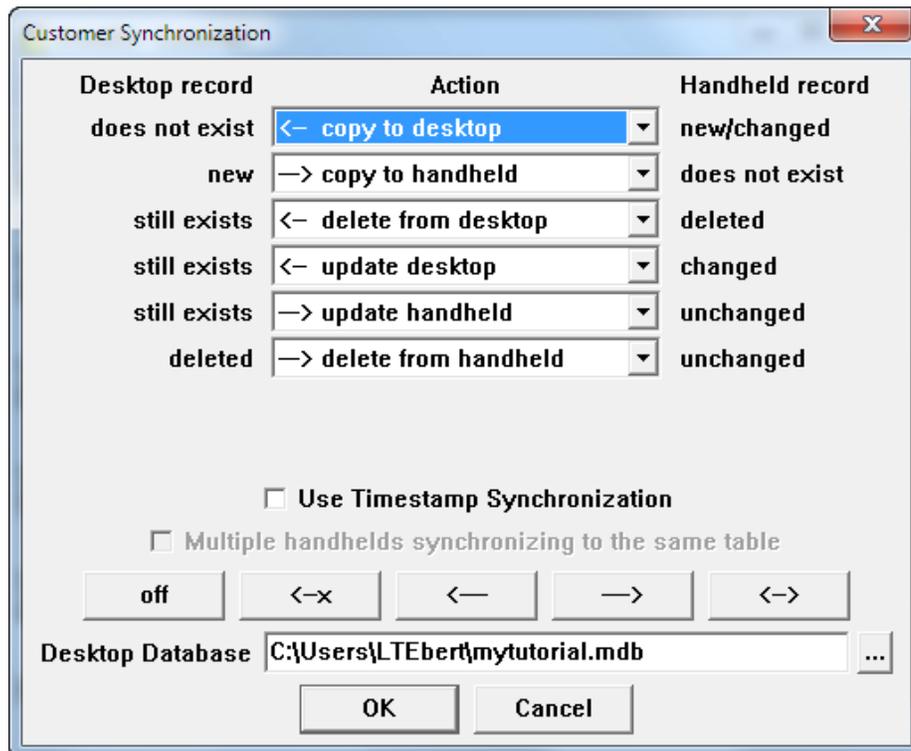
- ←x Copy new or modified records from the handheld table to the desktop PC table, and delete them from the handheld table (e.g., move records from the handheld to the desktop PC).
- ← Apply all changes to records in the handheld table to the desktop PC table. Changes in the desktop PC table are NOT applied to the handheld table.
- Apply all changes to records in the desktop PC table to the handheld table. Changes in the handheld table are NOT applied to the desktop PC table.
-  Publish. Replace all records in the handheld table with all records from the desktop PC table.
- ↔ Apply all changes to records in the handheld table to the desktop PC table, and vice-versa (e.g., all changes appear in both tables.)
- Apply a custom option that you specify (explained in Step 4).

Notice that DroidDB automatically selected two-way synchronization (<->) for the Phonebook table that you downloaded in the previous lesson.

4. You can use the radio buttons in the middle of the dialog box to apply a different option to a table highlighted in the Handheld Table listbox. The "Custom" button in particular lets you view specific actions associated with each of the basic options, and to fine-tune them to your specific requirements. Try it out:

- Highlight **Phonebook** in the Handheld Table list box, and click the **Custom** radio button (alternatively, you could have double-clicked **Phonebook**).

DroidDB displays the Phonebook Synchronization dialog box, which lists the specific actions currently specified for the Phonebook table (in this case, the actions associated with two-way synchronization). Experiment with the drop-downs in the Action column to view the different options available.



- Click **Cancel** to close the dialog box without saving any modifications you may have made. You are back at the Synchronize dialog box.
5. Click **Close** to close the Synchronize dialog box without starting synchronization.
  6. Close DroidDB by selecting **File > Close**.

You've now explored the synchronization options offered by DroidDB. The section, "Synchronizing Tables" (page 288) provides additional details for further fine-tuning them to your specific circumstances.

**Next:**

Quick Tour IV takes a closer look at two features for making your forms dynamic and easy-to-use: macros and events.

# Quick Tour IV: Getting Acquainted with Macros and Events

## Overview of Quick Tour IV

Macros and events are two of DroidDB's most powerful features. A macro is a series of commands that execute automatically in sequence. Macros are useful for automating tasks that require a number of steps. DroidDB offers an extensive library of commands that you can use as the building blocks of your macros. Events are actions that can trigger macros – for example, the form opens or the user changes a value in an input field.

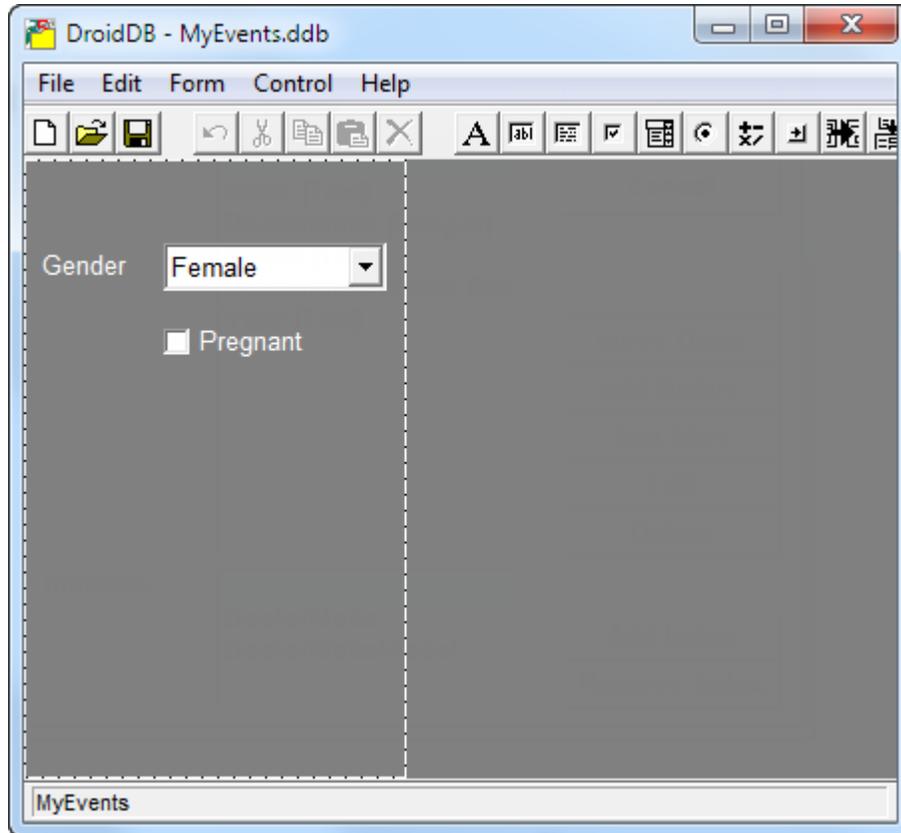
In this quick tour, you will build a sample application that uses a macro and an event to modify the form as the user interacts with it. Controls are displayed or hidden dynamically in response to user choices.

This tour consists of the following lessons:

- Quick Lesson 1: Creating the Form
- Quick Lesson 2: Planning the Macro
- Quick Lesson 3: Opening the Macro Builder
- Quick Lesson 4: Creating a Command to Hide a Control
- Quick Lesson 5: Creating a Command to Show a Control
- Quick Lesson 6: Adding a Skip Command with an “If” Expression
- Quick Lesson 7: Attaching the Macro to a Control Event

## Quick Lesson I: Creating the Form

Before you create a macro, you build the application's form. This sample form consists of just a few simple controls. You can use the illustration below as a guide for their placement.



### Note

Strictly for simplicity's sake, this sample application stores values in variables rather than a table. This is just to save the time and effort of setting up a table for this example.

1. Connect your Android device to your desktop PC.
2. On the desktop PC, click the Windows **Start** button, then **All Programs > SYWARE DroidDB > DroidDB**.
3. When DroidDB asks you to pick an application, select **Create new application**, enter the name **MyEvents**, and click **OK**.
4. Select **File > New Form**.
5. When DroidDB asks whether the form has an underlying table, for the purposes of this tutorial select **No**. (Again, this has nothing to do with macros...this is just to keep the sample as simple as possible. In most cases, your forms will store records in a table.)
6. Create the Gender Drop-Down List control:
  - Select **Control > Drop Down**
  - In the Select Column dialog box, select **@var(0)** and **Text**. Click **OK**.

### Notes

@var(n) is a global variable you can use to hold a value for processing when you don't need or want to write the values to a table.

- In the Drop Down Properties dialog box, click **Add After**.
  - In the Option Specification dialog box, enter **Female** in the Name field and select the **Default** option so that it is checked. Click **OK**.
  - Click **Add After** and enter **Male** in the Name field. Click **OK**.
  - Leave the remaining values in place and click **OK** to close the Drop Down Properties dialog box.
  - Resize and drag the control into place.
7. Add a label:
- Select **Control > Label**.
  - Enter **Gender** in the Text field, and click **OK**.
  - Resize and drag the label into place.
8. Create the Pregnant checkbox:
- Select **Control > Checkbox**.
  - In the Select Column dialog, select **@var(1)** and click **OK**.
  - For Text, replace the default value with **Pregnant** and click **OK**.
  - Resize and drag the control into place.
9. Select **File > Save**.

## Quick Lesson 2: Planning the Macro

Now you're ready to create the macro. But first, it's always a good idea to plan what you want your macro to accomplish, and how, before you actually build it.

Obviously, the Pregnant checkbox is only relevant when the user selects Female in the Gender drop-down menu. The macro you are going to write will hide the Pregnant checkbox if the user selects Male, and show it if the user selects Female. You can express these logical steps on paper using language similar to the DroidDB commands that make up a real macro, as follows:

- Step 1: Check the user's selection in the Gender drop-down. If it's Male, hide the checkbox (that is, go to the next step). If it's Female, show the checkbox (that is, go to step 4).
- Step 2: Hide the Pregnant checkbox.
- Step 3: Return from the macro – that is, stop here.
- Step 4: Show the Pregnant checkbox.
- Step 5: Return from the macro.

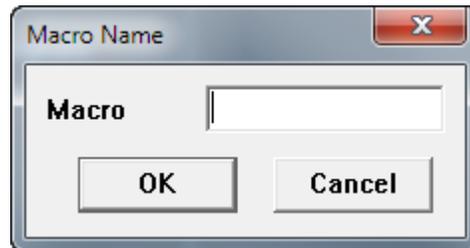
In the next lessons, you will use the Macro Builder to build these steps using DroidDB commands.

## Quick Lesson 3: Opening the Macro Builder

The Macro Builder is a convenient development environment that helps guide you through the process of writing, editing, and managing macros.

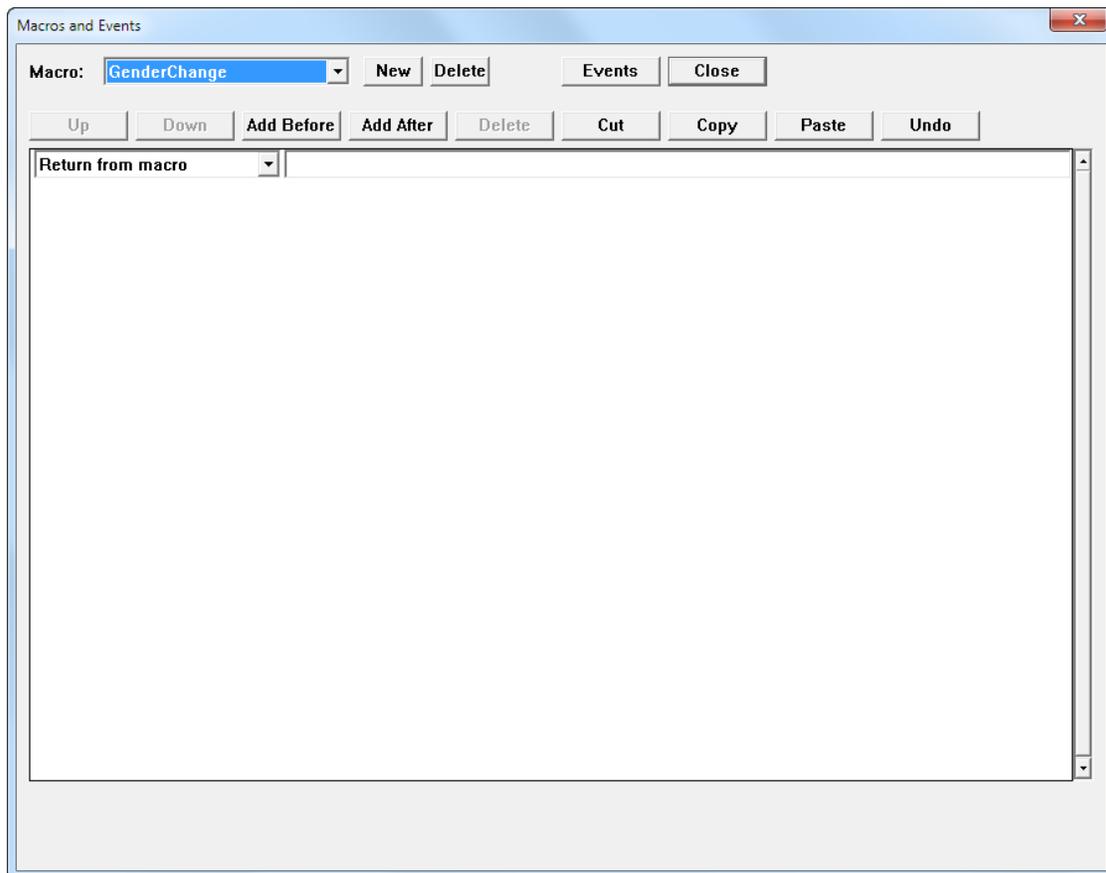
1. Select **Edit > Macros / Events**.

DroidDB displays the Macro Name dialog box.



2. Enter **GenderChange** and click **OK**.

DroidDB opens the Macro Builder window.



The Macro Builder has two buttons for managing macros: **New** and **Delete**. The **Events** button lets you associate a macro with some types of events. **Close** closes the Macro Editor. The remaining buttons let you organize the commands that make up a macro script in the scripting window (the large white area).

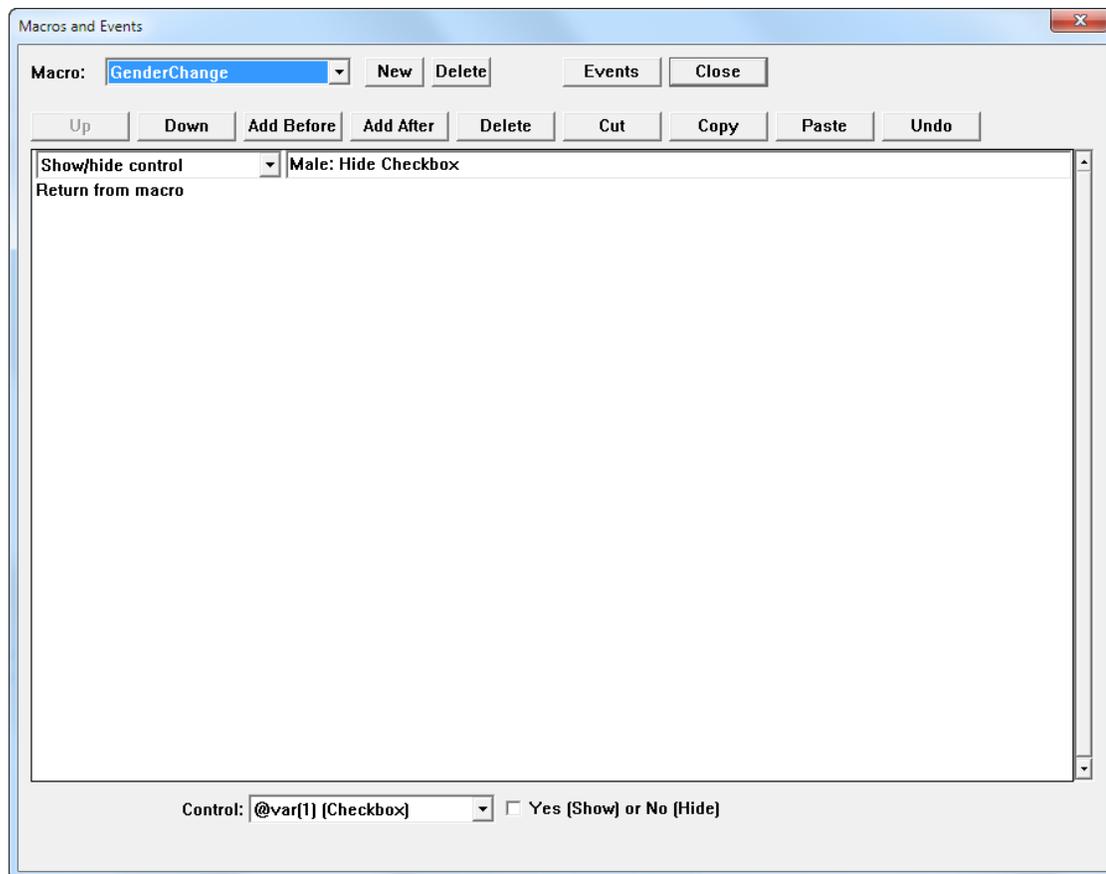
## Quick Lesson 4: Writing a Command to Hide a Control

For each step in your macro, you simply select one of DroidDB's pre-defined commands and specify the options specific to it. In this lesson, you will create a step that hides the Pregnant checkbox using the "Show/hide control" command.

1. In the Macro Builder, click **Add Before**.  
DroidDB inserts a new command at the beginning of the script.
2. Click the newly added command to display the command menu. Select **Show/hide control**.
3. At the bottom of the Macro Builder window, DroidDB displays the options for "Show/hide control" commands:
  - For control, select **@var(1) (Checkbox)**
  - Leave the "Yes(Show) or No(Hide)" option unchecked -- meaning No(Hide)
4. In the comments field to the right of the command, enter **Male: Hide Checkbox**.

### Note

Comments do not affect how the command runs. Use them to provide brief descriptions of what each command does, in case you need to modify or debug the macro later.



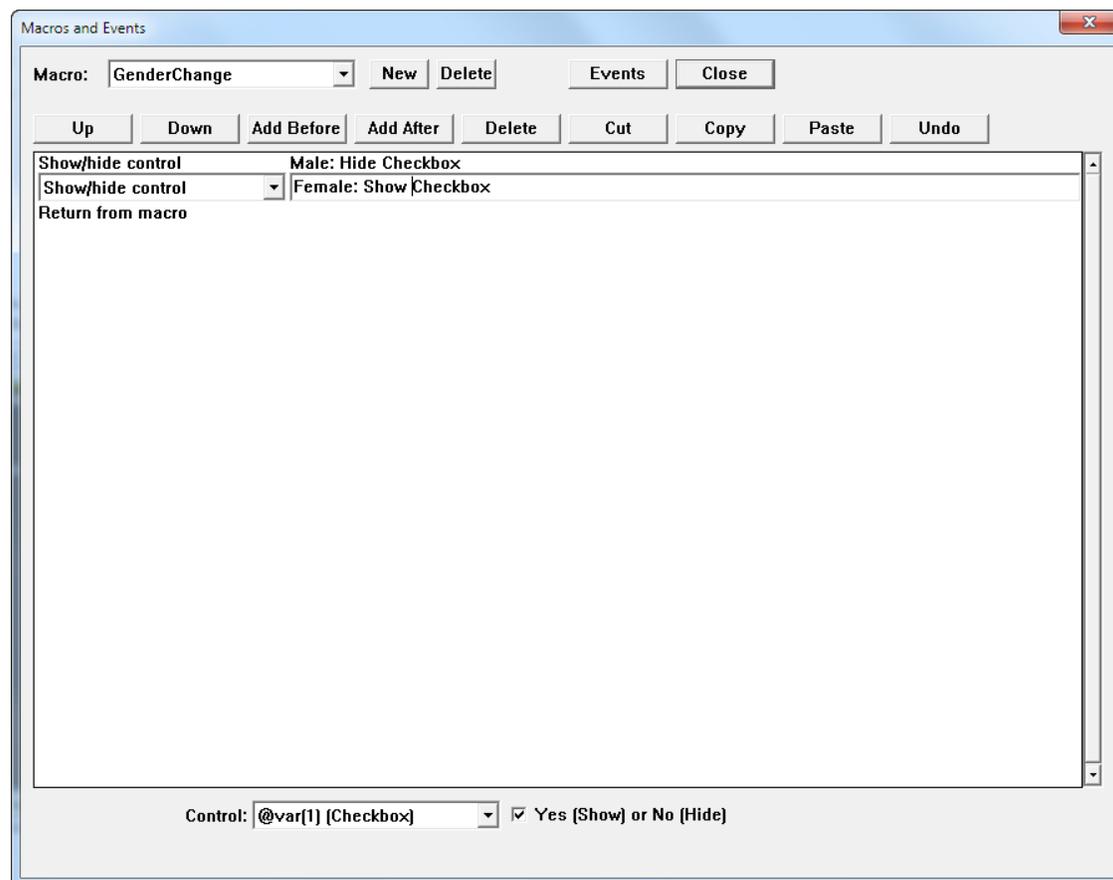
You've just created a command that, when executed, hides the Pregnant checkbox.

## Quick Lesson 5: Writing a Command to Show a Control

Now, create a second Show/hide command; this one to show the checkbox. This lesson also demonstrates the Macro Builder's copy/paste feature.

1. With the 'Show/hide control' command you just created still selected, click **Copy**.
2. Click **Paste**.  
The Macro Editor inserts the new command below the original.
3. Update the options for the new command:
  - For Control, select **@var(1) (Checkbox)**.
  - Check the **Yes(Show) or No(Hide)** option -- meaning Yes(Show).
4. In the comments field to the right of the command, enter **Female: Show Checkbox**.

At this point, your macro should look similar to the one shown below:



## Quick Lesson 6: Adding Conditional Branching Logic

In the simplest macros, the commands execute one after the other in the order they appear in the script. But sometimes you want a more complicated flow, such as conditional branching in which the macro executes one set of commands if a condition is met and another if it is not.

In DroidDB macros, you can accomplish this by using an “if” expression in a Skip command.

In this lesson, you’ll create commands that correspond to Step 1 and Step 3 in the plan on page 73.

1. With the second Show/hide command still selected, click **Add Before**.
2. Click **Up** to move the new command to the beginning of the script.
3. Select **Skip** from the drop-down list of available commands.

DroidDB displays the Num field at the bottom of the Macro Builder window.

4. In the Num field enter the following expression:

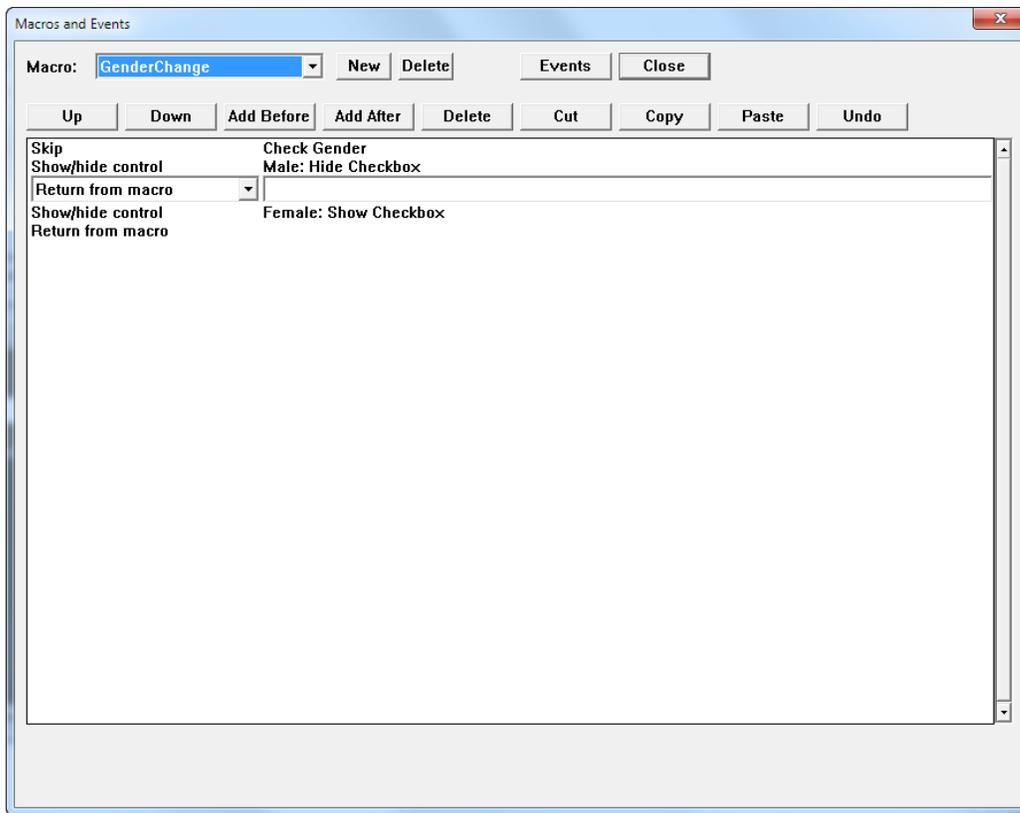
**if @var(0) = 'Male' then 0 else 2**

This expression instructs DroidDB to execute the next command in the macro if the value in the global variable @var(0) is 'Male'; otherwise, to skip two commands.

5. In the comments field to the right of the Skip command, enter **Check Gender**.
6. Select the first Show/Hide control, and click **Add After**.
7. Select **Return from macro** from the command drop-down list.

This command stops the macro after it hides the checkbox. Otherwise, it would continue running and display the checkbox.

You’ve now completed all of the commands in the sample macro. Your macro script should look like the one shown in the following illustration.



8. Close the Macro Builder by clicking the **Close** button near the top right side of the window.
9. To save the macro, you must save the form. Select **File > Save** from the development environment's menu bar

**Important!**

Macros are saved as part of the form. If you close the DroidDB development environment without saving the form, your macro will be lost.

## Quick Lesson 7: Attaching the Macro to a Control Event

Of course a macro is useless without a way to start it. In this lesson, you will set it up so that the GenderChange macro runs whenever when the user changes the value in the Gender drop-down control.

### Note

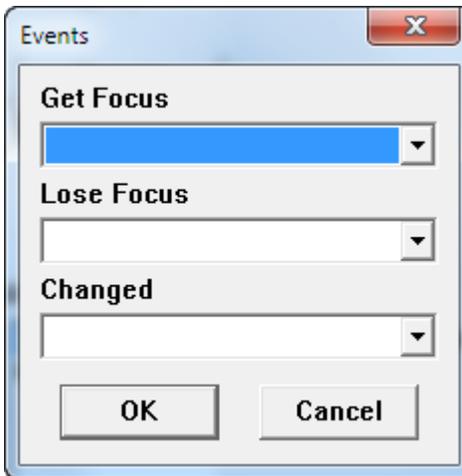
There are many other ways to start a macro. They are described in detail in the section, Macros and Events, later in this guide.

1. In the form, right-click or double-click the Gender drop-down list control.

The Drop-Down Properties dialog box opens.

2. Click the **Eventz** button.

The Events dialog box opens.



3. For Changed, select **GenderChange**. Click **OK**.
4. Click **OK** to close the Drop-Down Properties dialog box.
5. Select **File > Save**.

You've now completed the sample macro. Be sure to test it on the Android device.

### Next:

This quick tour has provided just a small glimpse of the capabilities of macros and events. The possibilities are described in more detail in the reference section beginning on page 212. In addition, be sure to check out the SYWARE DroidDB website, in particular the Tip of the Month section, <http://www.droiddb.com/totm/>, for sample macros and ideas.



## **Part II: Creating and Managing Your Own DroidDB Applications**

# Creating and Managing DroidDB Applications

## Overview of Part II

If you created the sample application in Quick Tour II, you've already mastered the basics of creating DroidDB applications. This section contains a more detailed description of each step involved, and presents the many additional features and controls not covered in the Quick Tour.

The following topics are discussed:

- DroidDB Development Environment
  - ◇ Starting and Exiting
  - ◇ Device screen size
  - ◇ Menu commands, toolbar buttons, and shortcut keys
- Getting Started
  - ◇ Overview of the steps required to create an application
  - ◇ Planning an application
  - ◇ Building Multi-Table/Multi-Form Applications
- Creating and Managing Tables
  - ◇ Creating a table
  - ◇ Modifying a table
  - ◇ Deleting tables
- Global Variables
  - ◇ About using global variables to store temporary values
- Creating and Managing Forms
  - ◇ Beginning a New Form
  - ◇ Modifying an Existing Form
  - ◇ Deleting a Form
- Customizing Forms
  - ◇ Specifying the form's size and color
  - ◇ Creating a title
  - ◇ Adding tabs
- Adding Controls to the Form
  - ◇ Edit boxes
  - ◇ Note boxes
  - ◇ Labels
  - ◇ Checkboxes
  - ◇ Radio buttons
  - ◇ Drop-down lists
  - ◇ Calculated fields
  - ◇ Command buttons

- ◇ Scribble boxes
- ◇ Image controls
- ◇ Jump buttons
- ◇ Lookups
- ◇ Grid controls
- Modifying Control Properties
  - ◇ Changing control properties
  - ◇ Specifying control font properties
  - ◇ Changing properties for multiple controls
- Arranging Controls on the Form
  - ◇ Turning off snap-to-grid
  - ◇ Aligning and sizing controls
  - ◇ Cutting, copying, and pasting controls
  - ◇ Deleting controls
- Defining Record Sorts and Searches
  - ◇ Specifying an Initial Sort/Search Ordering
  - ◇ Indexing Columns
  - ◇ Creating Pre-defined Filters for Record Display
- More Form Customization
  - ◇ Creating "About Box" text
  - ◇ Allowing inserts and deletes
  - ◇ Enabling auto recalc
  - ◇ Locking the form design
- Managing, Testing, and Distributing Applications
  - ◇ Saving an application
  - ◇ Opening and modifying an existing application
  - ◇ Converting a Visual CE application to a DroidDB application
  - ◇ Testing an application
  - ◇ Archiving and deleting applications
  - ◇ Creating distribution files (Business Edition Only)

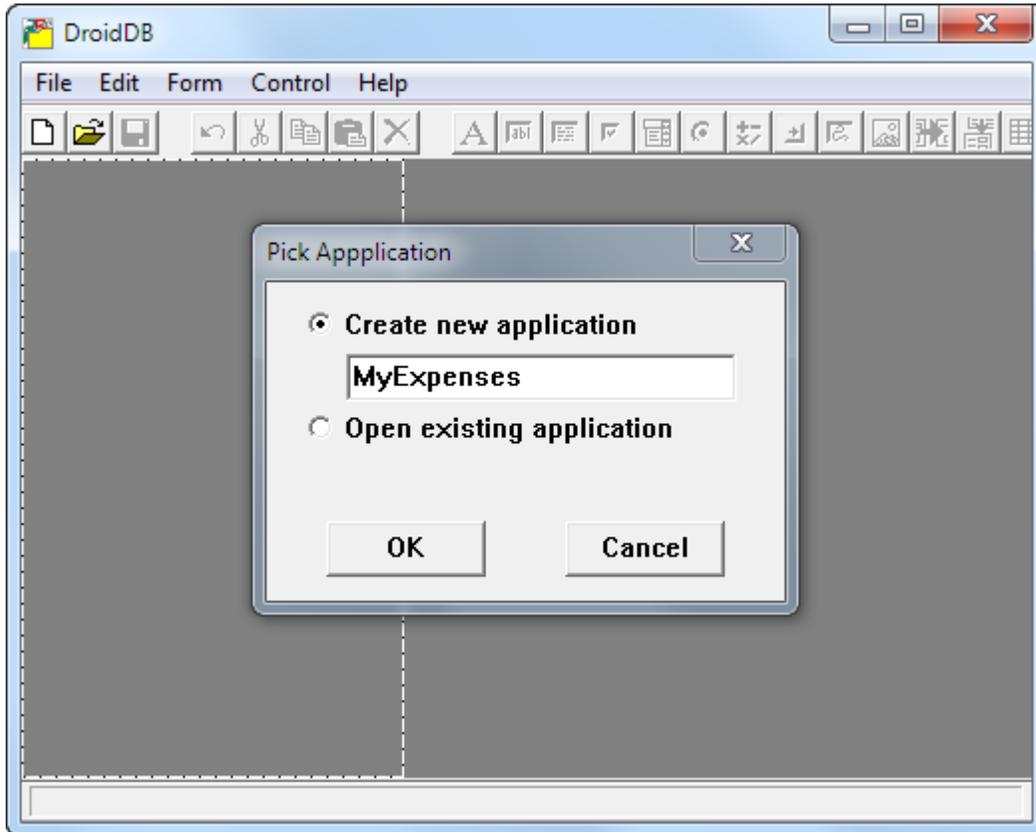
# DroidDB Development Environment

## Starting DroidDB's Development Environment and Beginning a New Application

All of your application development work is carried out in DroidDB's development environment on the desktop PC. Because DroidDB actually builds the application on the Android device's storage card as you work, the Android device must be connected to the desktop PC throughout your development session.

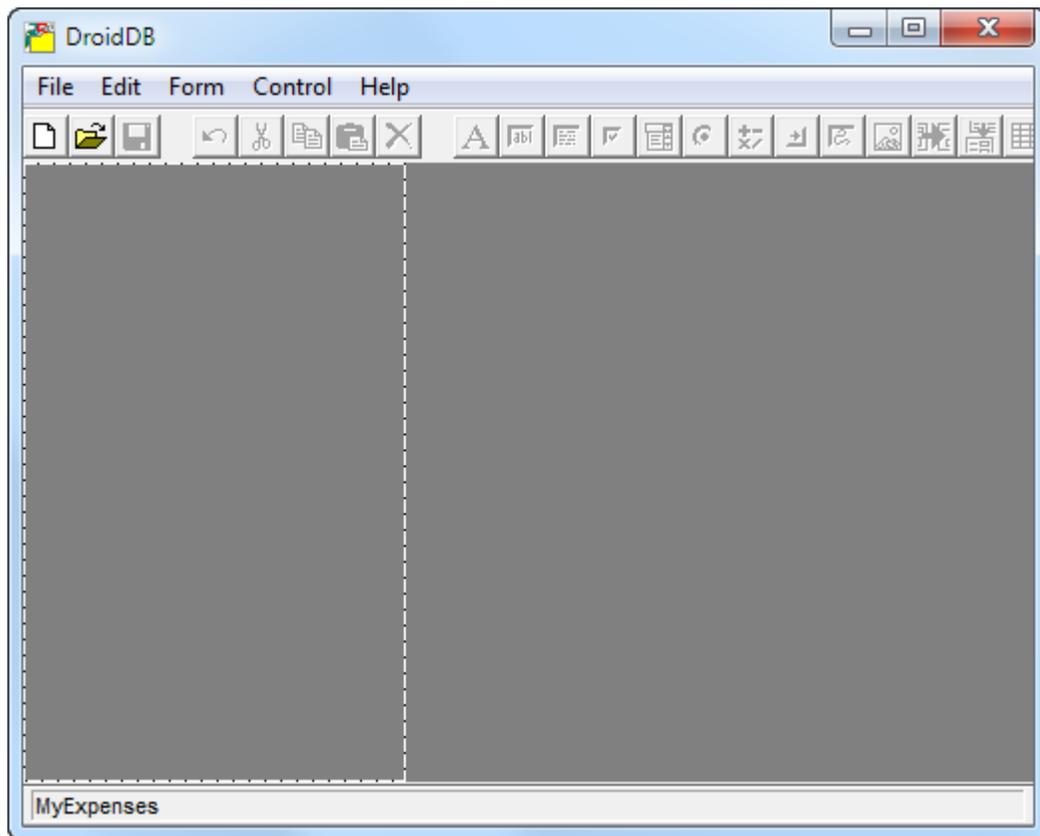
1. Connect your Android device to your desktop PC.
2. On the desktop PC, double-click the DroidDB icon to start DroidDB. Or, click the Windows **Start** button, then **All Programs > SYWARE DroidDB > DroidDB**.

DroidDB asks if you want to open or create an application.



3. Make your choice as desired:
  - If you select **Create new application**, you must give your application a name. DroidDB will create an application folder, with the name you supplied, in the top-level folder on the Android device storage card. Note: The name cannot contain spaces.
  - If you select **Open existing application**, DroidDB asks you to choose the application.

DroidDB displays its development environment window. The name of the application with which you are currently working is displayed in the status bar in the window's lower-left corner.



#### Note

Once you create or select an application, all of your work – creating tables, building forms, specifying synchronization settings – is done within the scope of that application. Applications are self-contained and work independently of one another. Applications do not share databases, tables, forms, or synchronization settings. If you want to work on a different application, you must exit and restart the DroidDB development environment, and then pick or create the other application.

DroidDB's development environment window has five menus: File, Edit, Form, Control, and Help. You can select menu options by clicking them with the mouse pointer or by pressing the shortcut keys. (To enable the shortcut keys, press the Alt key on your keyboard.)

The toolbar buttons provide a faster way to activate the most frequently used menu options.

The dotted outline in the development window indicates the screen size of the Android device. This provides a handy reference as you lay out the controls on the form. DroidDB automatically detects the type of Android device connected and displays the default screen size for that type. If you wish to lay out a form for a larger or smaller display, you can adjust the default using the instructions in the topic "Device Screen Size" on page 86.

DroidDB's default form size matches the screen size. If desired, you can specify a different height for your form. (See "Form Size" on page 113 for additional information.)

The status bar in the lower-left corner provides helpful feedback as you build the application.

## Changing the Device Screen Size

Typically, you build forms for the Android device that is connected to the desktop PC. DroidDB automatically detects the screen size of the connected Android device and displays its outline in the development environment window for your reference as you position controls on the form.

On rare occasions, you may wish to build forms for an Android device (call it the “target device”) other than the one you will use for development (call that one the “development device”). While the development device will be connected during your development session, you want the dotted outline in the development environment window to reflect the screen size of the (not-connected) target device.

### Note

Before starting the procedure below, you must have run DroidDB on the Android devices at least once (to finish the installation of DroidDB on the Android devices).

To do this:

1. Connect the target device to the desktop PC.
2. Copy a file called `DroidDB_Info.ini` from the `/DroidDB` folder on the Android device to the `Windows` folder on the desktop PC.
3. Disconnect the target device from the desktop PC.
4. Connect the development device to the desktop PC.
5. Run DroidDB on the desktop PC.

You will see that, although the development device is attached, the dotted line reflects the screen size of the now disconnected target device.

### Tip

If you develop DroidDB applications for a number of different Android devices, you may want to create a “library” of `DroidDB_Info.ini` files, one for each device type. That way, when developing for a particular target device, you can simply retrieve the appropriate `DroidDB_Info.ini` from your library and copy it into the `Windows` folder on your desktop PC.

## File Menu Commands

Command	Button	Shortcut	Function
New Form		Ctrl+N	Begins the creation of a new form.
Open		Ctrl+O	Opens an existing form for viewing and/or modifying.
Save		Ctrl+S	Saves the form currently displayed in the development environment window.
Save As	(none)	(none)	Makes a copy of the currently displayed form and saves it with a new name. The new form read/writes data to the same table as the original form.
Delete Form	(none)	(none)	Allows you to delete a form.
Create Table	(none)	(none)	Begins the creation of a new table.
Modify Table	(none)	(none)	Allows you to add and index columns in an existing table belonging to the current application.
Delete Table	(none)	(none)	Allows you to delete one or more tables.
Download Table	(none)	(none)	Downloads a desktop PC table to the Android device and creates a DroidDB application for it.
*Synchronize	(none)	(none)	Allows you to synchronize tables on the desktop PC and Android device.
Handheld Name	(none)	(none)	Displays the name currently assigned to the attached Android device and enables you to specify a new name. (Each device must be named for multi-device synchronization.)
*Create Distribution Files	(none)	Ctrl+W	Starts a wizard to create installation files for distributing your application to other users.
Close		Ctrl+W	Closes the DroidDB development environment.

*\*Business Edition only*

*Note: All file commands work within the context of the current application. For example, if you create a new form, it will belong to the current application. If you want to work with a different application, close and reopen DroidDB, then pick or create the desired application.*

## Edit Menu Commands

Command	Button	Shortcut	Function
Undo		Ctrl+Z	Undoes last edit.
Cut		Ctrl+X	Cuts the currently selected control and saves it to the clipboard.
Copy		Ctrl+C	Copies the currently selected control and saves the copy to the clipboard.
Paste		Ctrl+V	Pastes the contents of the clipboard onto the development area. You can copy or cut controls from one application or form, and paste them into another.
Delete		Delete	Deletes all currently selected controls.
Select All	(none)	Ctrl+A	Selects all controls in the development area.
Property	(none)	(none)	Displays the properties dialog box for the currently selected controls(s) for modification.  Note: Double-click or right-click any control to display and/or modify its properties.
Filters	(none)	(none)	Enables you to create and save record filters as part of the form design.
Macros/Events	(none)	(none)	Opens the Macro Builder where you can specify a sequence of commands to be executed upon a single command or event.
Custom Colors	(none)	(none)	Add up to 16 custom colors to the standard palette. You can apply these colors to the form.
Align	(none)	(none)	Enables you to align selected controls with one another along their right, left, top, or bottom edges or their horizontal or vertical centers; or to make a group of selected controls the same height or width. See "Aligning and Sizing Controls on the Form," on page 177.

## Form Menu Commands

Command	Function
Title Bar	Allows you to enter text to appear in the application's title bar.
Tabs	Allows you to divide your form into tabbed pages.
Sort On	Allows you to specify an initial sort/search column that DroidDB uses to organize records and to enable record searches.
Form Size	Allows you to change the size of the form.
Lock	Allows you to lock the form design so that no one can alter it..
Color	Allows you to specify a background color for the form.
About Text	Allows you to specify the text that will be displayed in the About box.
Allow Inserts	Allows you to specify whether or not the user can add new records to the table. If checked, the user can add records; if unchecked, the user cannot add records.
Allow Deletes	Allows you to specify whether or not the user can add delete records from the table. If checked, the user can delete records; if unchecked, the user cannot delete records.
Auto Recalc	Allows you to specify when DroidDB updates values for calculated field, lookup, and grid controls. If this option is checked, DroidDB automatically recalculates the values whenever the user changes a value on the form. If this option is unchecked, the user must explicitly select the Option > Recalc command.

## Control Menu Commands

Command	Button	Function
Label		Creates a label control.
Edit		Creates a single-line edit control.
Note		Creates a multi-line edit control.
Checkbox		Creates a checkbox control.
Drop Down		Creates a drop-down list control.
Radio Buttons		Creates a group of radio button control.
Calculated		Creates a calculated field control.
Command button		Creates a command button.
Scribble		Creates a scribble box.
Image		Creates an image control.
Jump		Creates a jump button.
Lookup		Creates a lookup control.
Grid		Creates a grid control.

## Development Window Toolbar Buttons

Button	Name	Function
	New Form	Begins the creation of a new form.
	Open	Opens an existing form for viewing and/or modifying.
	Save	Saves the form currently displayed in the development environment window.
	Undo	Undoes last edit.
	Cut	Cuts the currently selected control and saves it to the clipboard.
	Copy	Copies the currently selected control and saves the copy to the clipboard
	Paste	Pastes the contents of the clipboard onto the development area. You can copy or cut controls from one application or form, and paste them into another.
	Delete	Deletes all currently selected controls.
	Label	Creates a label control.
	Edit	Creates a single-line edit control.
	Note	Creates a multi-line edit control.
	Checkbox	Creates a checkbox control.
	Drop Down	Creates a drop-down list control.
	Radio Buttons	Creates a group of radio button control.
	Calculated	Creates a calculated field control.
	Command button	Creates a command button.
	Scribble	Creates a scribble box.
	Image	Creates an image control
	Jump	Creates a jump button.
	Lookup	Creates a lookup control.
	Grid	Creates a grid control.

## Development Window Shortcut Keys

### File Functions

To do...	Press
Begin creation of a new form.	Ctrl+N
Open an existing form for viewing and/or modifying.	Ctrl+O
Save the form currently displayed in the development environment window.	Ctrl+S
Close the DroidDB development environment	Ctrl+W

### Edit Functions

To do...	Press
Undo last edit.	Ctrl+Z
Cut the currently selected control and save it to the clipboard.	Ctrl+X
Copy the currently selected control and save the copy to the clipboard.	Ctrl+C
Paste the contents of the clipboard onto the development area. You can copy or cut controls from one application or form, and paste them into another.	Ctrl+V
Delete all currently selected controls.	Delete
Selects all controls in the development area.	Ctrl+A

### Selecting and Arranging Controls

To do...	Press
Select multiple controls.	Ctrl while selecting; or left-mouse click and drag cursor across controls.
Deselect a single control in a selected group.	Ctrl while selecting
Turn off snap-to-grid.	Shift while dragging

## Exiting DroidDB's Development Environment

You can exit the DroidDB development environment window when you have finished developing an application, or at any point during application development.

1. Select **File > Close** or click the **Exit** button  on the title bar.

DroidDB asks if you want to save your changes.

2. Click **Yes** to save your changes, or **No** to exit without saving.

# Getting Started

## Steps in Creating an Application

The sequence for creating DroidDB applications is:

1. Plan the application.
2. Open the DroidDB development environment and create a new application.
3. Optional: Create a table to store the application's data.
4. Begin the form:
  - ◇ Specify the table that the form will use; could be no table
  - ◇ Create a title for the form
  - ◇ Specify form size and background color
  - ◇ Optional: Create and name tabbed pages
5. Create and place controls on the form. There are many control types to choose from, including edit boxes, check boxes, command buttons, and several more.
6. Arrange the controls on the form.
7. Optional: Define record sorts and searches:
  - ◇ Specify an initial sort/search ordering for record display
  - ◇ Create filters for record display
8. Optional: Customize the form:
  - ◇ Specify text for the About box
  - ◇ Allow/prevent users from inserting/deleting records
  - ◇ Enable Auto Recalc for calculated field values
  - ◇ Prevent modification to the form design
9. Optional: Add a macro program
10. Save the application
11. Test the application
12. Optional / Business Edition Only: Set up synchronization between the DroidDB application table and Microsoft Access database or other ODBC-enabled data source on the desktop PC or remote server.

Each step is described in detail in the following topics and sections.

### Tip

If you have a Visual CE application for Windows CE devices, it is very easy to turn it into a DroidDB application for Android devices. Please refer to page 193.

## Planning the Application

If you want to create a DroidDB application that can be used far into the future, spend a little time planning your application before starting to build it.

Be sure to consider:

- What data will your application collect.
- Should the application's data be stored in one or a number of related tables, or perhaps a table is not necessary at all.
- The types of controls you will use and where you will place them on the form.

### Note

The form height can exceed that of the Android device screen. See "Form Size" on page 113 for further information. Also, you can specify the size of the Android device screen that is indicated by the dotted lines in the DroidDB development environment window. See "Device Screen Size," on page 86 for further information.

- Whether the labels and captions you plan to use on the form are self-explanatory (meaningful to anyone beside yourself)
- Whether a particular control will be read-only (display only - no user input) or read/write (allowing user input)
- Whether a particular control will have a default value
- Whether input will be required for a particular control
- Whether you want records organized based on a particular control value (sorting)

In general, it is better to keep edit boxes and note boxes to a minimum. Wherever possible, use drop-down lists, radio buttons, and checkboxes instead, because they consume less storage space and save the user from typing.

## About Building Multi-Table/Multi-Form Applications

As you work with DroidDB, you'll find that it enables you to build sophisticated applications consisting of any number of related tables and forms. This is especially useful if your application collects data about multiple related entities.

For example, say you have a table of ORDERS and a table of ITEMS. Each ORDER record has multiple, corresponding ITEM records, and records in both tables have a column called ORDER\_ID which relates the ORDER records to the ITEM records.

A DroidDB application can make use of this relational characteristic in the following ways:

- On the ITEM form you can have a button that, when pressed, launches the ORDER form and positions at that item's order.
- On the ITEM form, you can have a lookup control that displays information from the corresponding ORDER record (like date of the order, customer, etc.)
- On the ORDER form, you can have a grid that displays the items associated with that order. If the user does a double long-press on a particular item, DroidDB will launch the ITEM form and display that item.
- In the ITEM form, you can have a dropdown of possible items and this dropdown can be populated with data from another table, such as an INVENTORY table.
- (Business Edition only) The DroidDB synchronizer can synchronize both tables to the desktop.
- DroidDB can generate new, unique ORDER\_ID's when creating new ORDER records. In addition, DroidDB can automatically set an ITEM record's ORDER\_ID when a new ITEM record is created.

You can do all the above by just creating controls and setting their properties. No coding is needed.

### About Maintaining Secondary Tables

Every application has one form that opens automatically when the user starts the application on the Android device. (The first form you create for the application is automatically this primary form.) That means there is no direct way for users to open an application's secondary forms and add data to their tables. In that case, you could add a "Run Form" command button on the primary form that users could touch to open the secondary form. Or, if you have the DroidDB Business Edition, you could maintain a version of the secondary table on the desktop PC or remote server and update its Android device counterpart via synchronization.

### Multiple Applications Writing to the Same Tables

Applications work independently of one another; two applications cannot share a database, tables, or forms. If you find that you have two applications that need to share a database, combine them into a single application and create a table-less "switchboard" form to launch one or the other. For example, you might have an inventory control application with one form that adds inventory items to an Inventory table and another one that deletes inventory items from the same table. In that case, the application's primary form might consist of just two command buttons, one to open the "Add to Inventory" form and another to open the "Remove from Inventory" form.

# Creating and Managing Tables

## Creating Tables

Your first step when building a DroidDB application is almost always\* to create a table.

You can think of a table as a structure, internal to the computer, that stores a form's data and organizes it in rows and columns.

Rows store the records that users create when they work with your DroidDB form. For example, if your form tracks expenses, each row stores one expense record. A table can have any number of rows (records), subject to the limitations of the Android device.

Each column stores values for one category of information; in other words, for one field in the records. For example, one column might store expense dates, while another stores amounts, and another the descriptions.

When you create the table for your form, you specify a column for each category of information you want to store. You specify a name for the column, and the type of data it will store (text, integer number, floating point number, date/time, memo, or yes/no).

When you eventually create the form (the user interface), you create one control on the form for each column in the table. The control makes it possible for users to view, add, modify, and delete values in the table column. You can also have calculated field controls that calculate values and write them to a table column.

There are two ways to create a table to be used by a DroidDB application:

- Create the table using DroidDB (page 98). You would typically use this method when the table will be used on Android devices only.
- Download the table and data from an ODBC-enabled application on the desktop PC, such as Microsoft Access (page 101). You would typically use this method when you want to keep data in the desktop/server table synchronized with one or more Android devices.

The following two topics describe each approach.

### Tip

You can also recreate a lost table (page 103) and create a table and its forms from a Visual CE application (page 193).

### \*Note

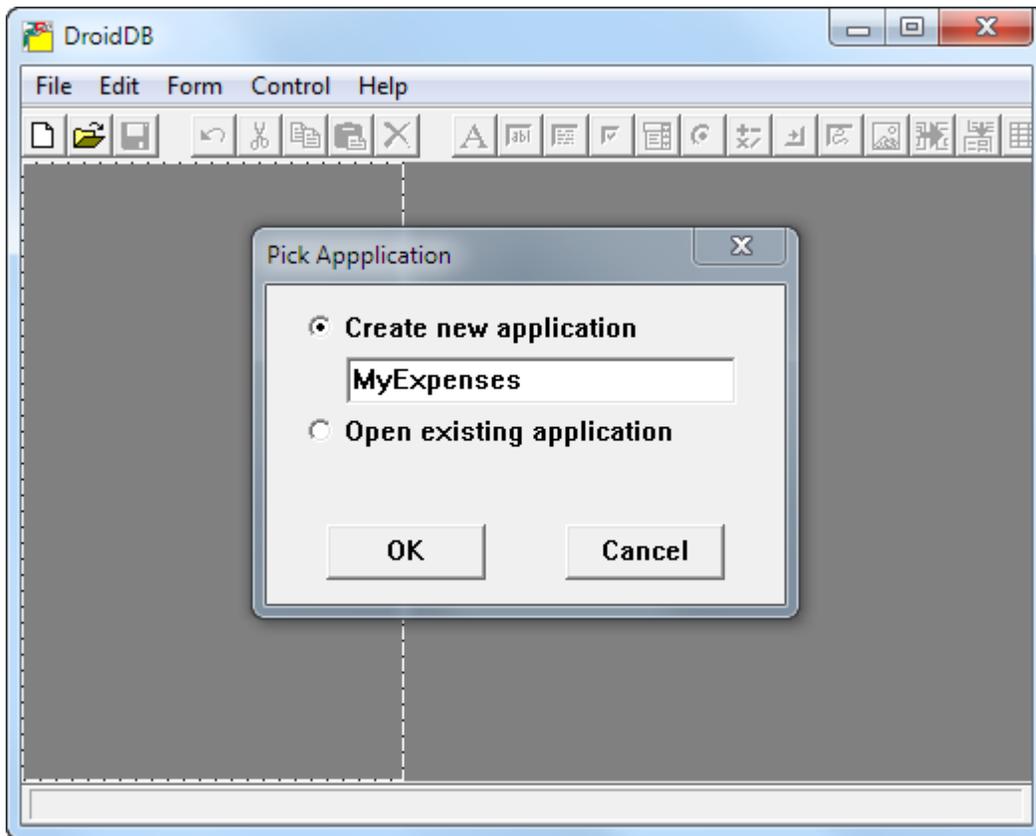
You can create controls that read and write values to global variables, rather than to table columns. This is useful if you have designed a process that requires temporary values. In rare instances, you may even want to create a form that consists entirely of controls that read and write values to variables, in which case you would not create a table for the form at all. Refer to the topic, "About Using Global Variables to Store Temporary Values," on page 107 for more information.

## Creating a Table Using DroidDB

Using DroidDB on the desktop PC, you can create a table that is initially empty. You can then use a DroidDB form you design to create records in the table.

Before you begin creating a new table, put some time into planning it carefully. Once a table is created, the only modification you can make to it is to add columns to the end of the column order.

1. Connect your Android device to your desktop PC.
2. Start the DroidDB development environment on the desktop PC, then create or open the DroidDB application to which the table will belong, as described on page 84.

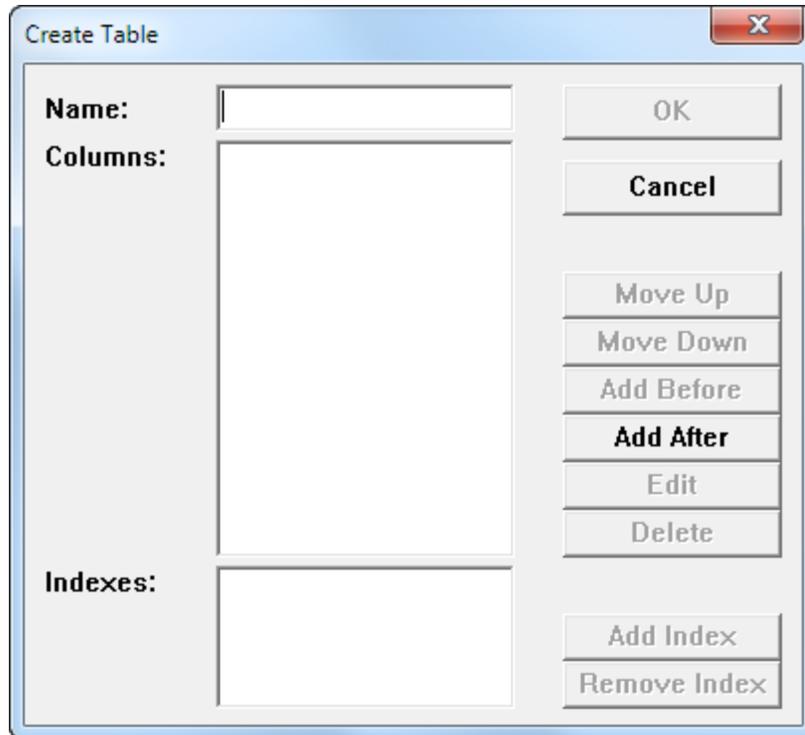


**Note**

A table can belong to just one application, but one application can have multiple tables.

3. Select **File > Create Table**.

DroidDB displays the Create Table dialog box.



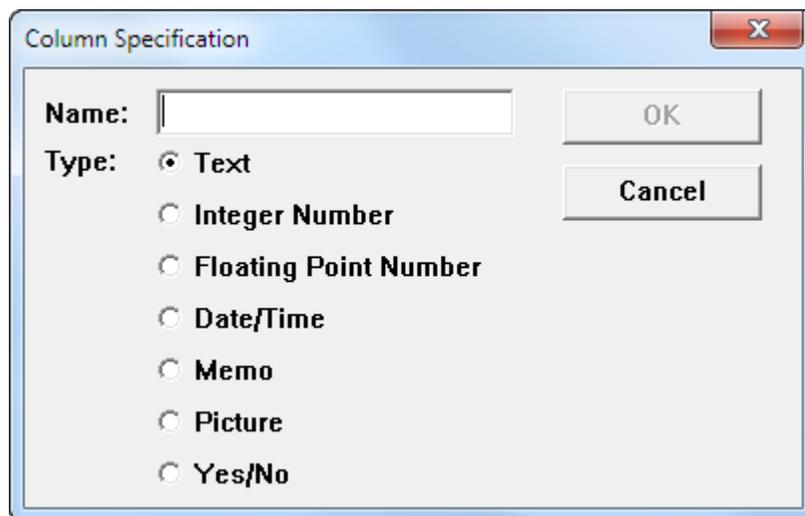
4. Enter the name of the table in the Name field.

**Note**

The table name and application name may, but need not be, the same.

5. Click the **Add After** button.

DroidDB displays the Columns Specifications dialog box.



6. Enter the name of the column in the Name field. Try to use a name that is meaningful and short.

**Note**

To facilitate synchronization, avoid names that contain special characters such as spaces, commas, or dashes, or that begin with a digit. (A message box appears if the name you choose

could cause problems.)

7. Specify the type of data that the column will store by clicking the corresponding radio button:
  - **Text** Small amounts of text (less than 256 characters); if you'll have larger amounts of text, use Memo instead
  - **Integer Number** Whole numbers
  - **Floating Point Number** Numbers with digits to the right of the decimal point
  - **Date/Time** Dates, times, or both
  - **Memo** Large amounts of text (more than 255 characters)
  - **Picture** Scribbles, photos, or other graphics
  - **Yes/No** Also referred to as "Boolean"
8. Click **OK** to save your entries and return to the Create Table dialog box.
9. Click the **Add After** button (or **Add Before**, if you want this column to precede the column you already entered).
10. Continue specifying columns as described in Steps 6 through 8 until you have defined all of the table's columns.

**Note**

If you need to change your entries for a column, click the column (to highlight it), then click the **Edit** button. To remove a column, click the column, then click the **Delete** button. If you want to change the order of the columns, click a column, then click the **Move Up** or **Move Down** button.

11. Indexes are used for record sorting and searching. If you intend to use a column in the application for record sorts, searches, lookups, grid controls, or jumping to another table, you should index the column now:
  - Click the **Add Index** button
  - Select the column to index, and click **OK**

**Note**

It is not necessary to index a column unless you want to specify an initial sort/search ordering (see "Specifying an Initial Sort/Search Ordering," on page 180), create a jump button (see "Creating a Jump Button," on page 161), use the column as a lookup in another table (see "Creating a Lookup," on page 165), or create a grid control (see "Creating a Grid Control," on page 168). DroidDB supports up to four indexes per table at a time.

12. When you have finished specifying columns, click **OK** in the Create Table dialog box.

DroidDB creates an empty table in the database in the application's folder on the Android device. DroidDB's status bar (at the bottom of the development environment window) now displays the name of the application and the name of your new table.

13. DroidDB asks if you want the Form Wizard to create a form for this table:
  - Click **Yes** if you want the Form Wizard to automatically create a control for each column in the table, including labels that correspond to the columns' names. The Wizard arranges the controls along the left side of the development window. You can then quickly rearrange the layout and change the properties of any of the controls to meet your needs, if desired.
  - Click **No** if you want to create each control from scratch (as you did in Quick Tour II). Instructions for creating controls is provided beginning on page 122.

## Downloading a Table from an ODBC-Enabled Application on the Desktop PC

If you have a table on the desktop PC that was created with an ODBC-enabled application—such as Microsoft Access or Excel—you can download it (and any records it contains) to the Android device, and quickly create a DroidDB application for it. For Business Edition users, the download procedure also includes optional steps for establishing synchronization between the desktop PC table and the table on one or more Android devices.

### Note

The Android device must be connected to the desktop PC throughout the download procedure.

1. (Perform this step *if you have the Business Edition and if you plan to synchronize the tables.* Otherwise, go to step 2.) In order for synchronization to work, the first column of the desktop table must be an "OID" column. (DroidDB automatically adds a corresponding column to the table on the Android device and uses the columns to keep records in the two tables in sync.) Before starting the download procedure, open the table in the desktop PC application that was used to create it (e.g., Microsoft Access). Add a new first column to the table with the following specifications:

- Field name: **OID**
- Data Type: **Number**
- Indexed: **No**

The value stored in the OID column for each existing record must be zero or null (blank). Save your changes and close the application.

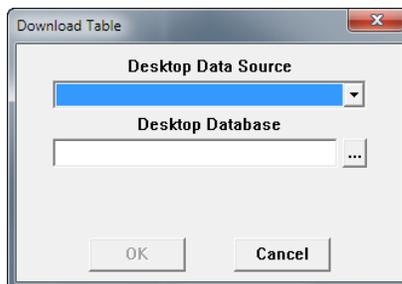
2. On the desktop PC, double-click the DroidDB icon to start DroidDB. Or, click the Windows **Start** button, then **All Programs > SYWARE DroidDB > DroidDB**.
3. Create or open the application to which the table will belong.
4. If you plan to synchronize the desktop PC table with multiple Android devices, each Android device – including the currently attached development device – must be uniquely named. To name the currently attached device, select **File > Handheld Name** and supply a name. You need do this just once for each Android device, not for each application.
5. Select **File > Download Table**.

Business Edition Users: DroidDB asks if you want to establish synchronization between the tables.

- Click **Yes** if you plan to maintain data in both the Android- and desktop-versions of the table and want to share updates between them.
- Click **No** if you simply want to download the table to the Android device (if you click No now, you can always set up synchronization later).

DroidDB displays the Download Table dialog box.

6. Select the database that contains the table you want to download:



- Click the arrow to the right of the "Desktop Data Source" field, and select the data source type
- Some datasources require auxiliary information. For example, an Access ODBC datasource may need the full pathname of the database:

- ◇ Click the Browse button , and navigate to the folder that contains the database
- ◇ Select the database and click **Open**.

- Click **OK** to close the Download Table dialog box. DroidDB displays the Select Table dialog box. Go to step 7.

If your application already has tables set up for synchronization and you specified that this table is to be synchronized, DroidDB assumes that you want to download the table from the same desktop PC database you used before. If that is the case, click **OK** and go to step 7.

But if you want to download the current table from a different desktop PC database, click **Cancel** to close the Download Table dialog box. Then:

- ◇ Select **File > Synchronize**. DroidDB opens the DroidDB Synchronize dialog box.
- ◇ Select the new **database** and other settings as desired (refer to page 289 for additional details). Click **Close** to save your new settings in the DroidDB Synchronize dialog box.
- ◇ Select **File > Download Table**. DroidDB displays the Download Table dialog box again, this time with the new database selected. Click **OK**.

DroidDB displays the Select Table dialog box.

7. Select the table you want to download, then click **OK**.
8. If you earlier indicated that you want to synchronize the tables, DroidDB asks if this table will be synchronizing to more than one handheld. Respond as desired. (For more information about this feature, please see "Synchronizing Multiple Handhelds on page 295).
9. DroidDB asks if you want the Form Wizard to create a form for this table:
  - Click **Yes** if you want the Form Wizard to automatically create a control for each column in the table, including labels that correspond to the columns' names. The Wizard arranges the controls along the left side of the development window. You can then quickly rearrange the layout and change the properties of any of the controls to meet your needs, if desired.
  - Click **No** if you want to create each control from scratch (as you did in Quick Tour II). Instructions for creating controls is provided beginning on page 122.

#### Note

*If any of the Android devices to be synchronized are "portable devices":* When connected to the desktop PC, an Android device may appear to Windows as a storage card (with a drive letter) or as a portable device (without a drive letter). The synchronization process for "portable devices" uses the "lite" version of mEnable server. That means that you, the form designer, must incorporate the "mEnable Synchronize" command (page 241) into the form – either as a command button that the user touches to initiate the synchronization, or as a step in a macro that initiates synchronization automatically. Note: Unlike the full version of mEnable for wireless synchronization, which is purchased separately, the 'lite' version for connected portable devices is provided as part of the DroidDB Business Edition.

If you are a Business Edition user, you can fine-tune the synchronization options to fit your needs (see pages 289 and 292).

## **Recreating a Lost Table**

If a table is lost, you can recreate its structure (but not its data) by simply opening a form (.ddb file) built over that table in either the DroidDB development environment or on the Android device.

## Adding Columns to an Existing Table

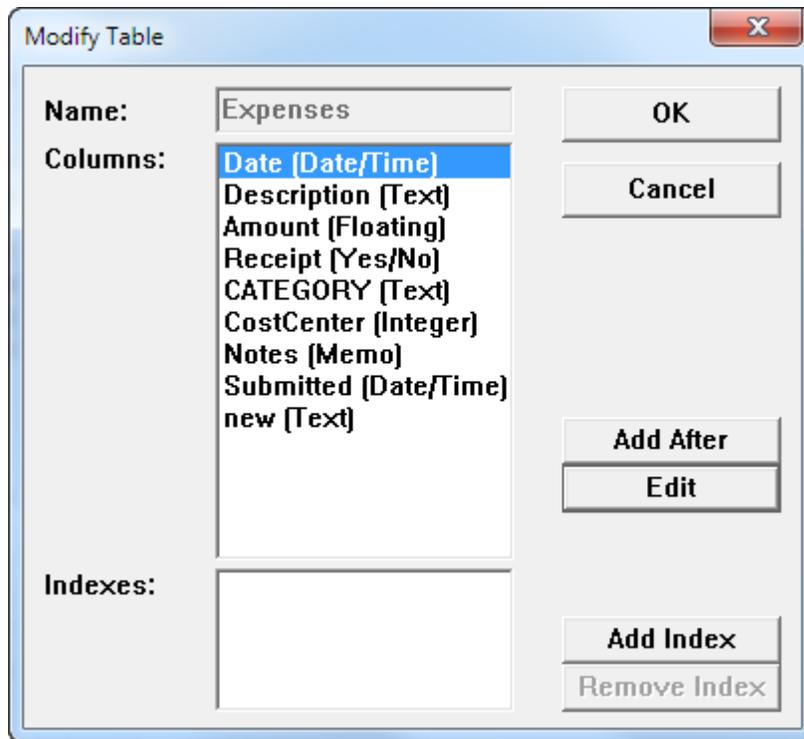
Using DroidDB's development environment on a desktop PC, you can add columns, but not delete or modify them.

### Note

Keep in mind that if you change the table structure, you may have to modify the forms that use it. Additionally, any changes to its structure can cause synchronization problems.

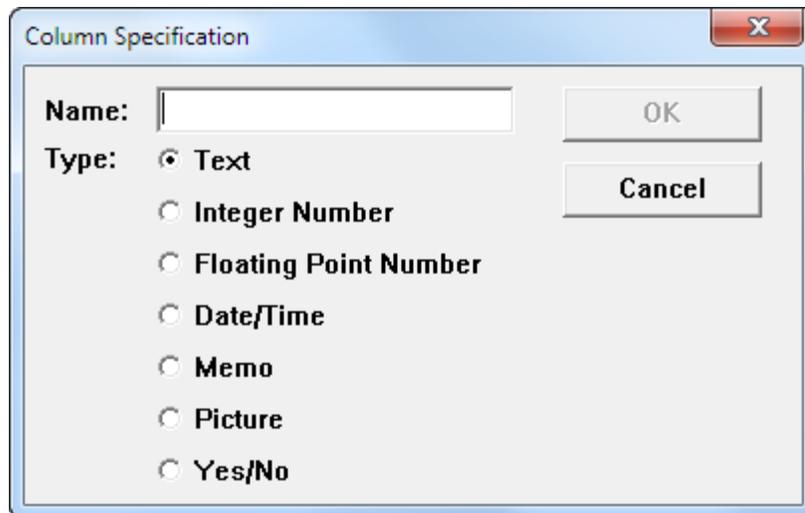
1. Working in DroidDB's development environment on the desktop PC, open the DroidDB application that contains the table you wish to modify.
2. Select **File > Open** and choose the form that is associated with the table.
3. Select **File > Modify Table**.

DroidDB displays the Modify Table dialog box.



4. Click the **Add After** button.

DroidDB displays the Columns Specifications dialog box.



5. Enter the name of the column in the Name field. Try to use a name that is meaningful and short. To facilitate synchronization, avoid names that contain special characters such as spaces, commas, or dashes, or that begin with a digit. (A message box appears if the name you choose could cause problems.)
6. Specify the type of data that the column will store by clicking the corresponding radio button. The datatypes are explained on page 100. Click **OK** to save your entries and return to the Create Table dialog box.
7. Continue adding columns as just described.

**Note**

If you change your mind about the name or datatype of a new column, select the column name in the Modify Table dialog box, and click the **Edit** button. The Columns Specifications dialog box opens. Make changes as desired.

8. Click **OK** to close the Modify Table dialog box.

**Note**

Your changes are saved immediately.

**Tip**

You can also use the Modify Table dialog box to index columns for sort and search orders. See "Indexing Columns," on page 181.

## Deleting Tables

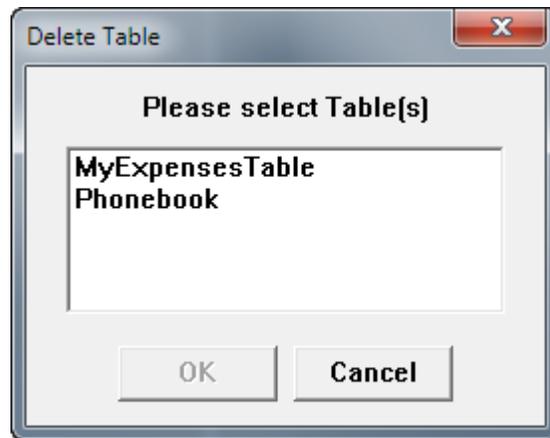
Deleting DroidDB tables from the Android device is easy. You can delete any number of tables at one time.

### Note

When you delete a table, DroidDB automatically deletes any forms built on it as well.

1. Working in DroidDB's development environment on the desktop PC, open the DroidDB application that contains the table(s) you wish to delete.
2. Select **File > Open** and choose the form that is associated with the table(s).
3. Select **File > Delete Table**.

DroidDB displays the Delete Table dialog box.



4. Click the names of the tables you want to delete, then click **OK**.

DroidDB asks you to confirm each selection, then deletes the specified table from the Android device.

# Using Global Variables

## About Using Global Variables to Store Temporary Values

Most DroidDB applications consist of a table that stores records, and a form made up of controls that read and write data to individual record fields (table columns).

It is also possible to associate a control with a DroidDB global variable, so that the control reads and writes a value to the variable rather than to a table column. This is useful when the value is required temporarily -- to facilitate a task such as jumping to another table, for example.

You can even create a form that consists entirely of controls that read and/or write values to variables, so that that the form has no underlying table at all.

Global variables are also useful for passing values between macro steps.

DroidDB stores values for up to 100 different global variables at one time: @var(0), @var(1) ... @var(99). The variables are global and persistent within the context of a DroidDB application -- meaning that assigned values are available to all forms belonging to an application, and that they persist until overwritten by new values -- even when the application is closed and the device turned off.

Variables can store any data type recognized by DroidDB except picture.

To associate a control with a variable, simply select the variable when you create the control (refer to the instructions for creating particular types of controls, beginning on page 122).

To use a global variable in a macro command, select the variable when specifying the command parameters. Refer to the command reference beginning on page 230.

For information about creating a form without a table, refer to the topic, "Beginning a New Form," on page 109.

### What Controls and Form Features Use Global Variables

The list below summarizes controls and other form features that read/write values to global variables:

#### Writes values to global variables:

- All controls that write to a table column, except Scribble controls
- Commands (Macros and Buttons)
  - ◇ Assign
  - ◇ Barcode
  - ◇ Communicate | GPS

#### Reads values from global variables:

- Controls
  - ◇ Dependent Drop-Down lists (filter value)
  - ◇ Grid control (key value in application table)
  - ◇ Jump (key value in "jump-from" table)
  - ◇ Lookup (key value in application table)

- ◇ Calculated field control

**Reads values from global variables:**

- Commands (Macros and Buttons)
  - ◇ Assign
  - ◇ Select Tab
  - ◇ Skip
  - ◇ Message box
  - ◇ Record | Search (search value)
- Pre-defined record filter

# Creating and Managing Forms

## Beginning a New Form

Before building a new DroidDB form, you typically\* create a table to store the form's records. If you have not already created the table, please refer to the previous section, "Creating and Managing Tables."

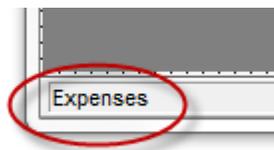
### \*Note

In almost all cases, you will create forms that read and write records to a table. In rare instances, you may need to create a form that reads and writes temporary values to global variables instead. In that case, you do not need to select a table before proceeding with the instructions below.

1. If it is not already open in the DroidDB development environment, open the DroidDB application to which the form will belong. If the form will read and write data to a table, that table must be in the current application's database.

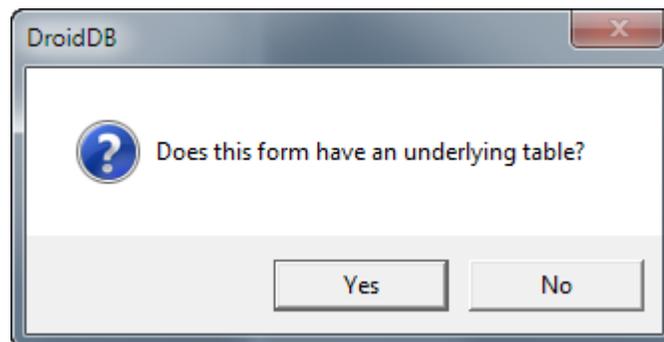
### Tip

The status bar in the lower left corner of the development environment displays the name of the currently open application.



- If another DroidDB application is open, close the development environment (File > Close).
  - Start the DroidDB development environment. When the Pick Application dialog box appears, select the application to which the form will belong.
2. From the menu bar, select **File > New Form**.

DroidDB asks if the form has an underlying table.



- Click **Yes** if the form will read and write records to a table. The Select Table dialog box appears.
- Select **No** if you will use global variables to store data associated with all controls on the form. If you select this option, you have completed this topic. You can now create your form using the instructions in the following sections, and you can add any type of controls to your form.

3. In the Select Table dialog box, click the table you want, then click **OK**.
4. DroidDB asks if you want the Form Wizard to create a form for this table:
  - Click **Yes** if you want the Form Wizard to automatically create a control for each column in the table, including labels that correspond to the columns' names. The Wizard arranges the controls along the left side of the development window. You can then quickly rearrange the layout and change the properties of any of the controls to meet your needs, if desired.
  - Click **No** if you want to create each control from scratch (as you did in Quick Tour II).

DroidDB displays the name of the specified table in the status bar at the bottom of the development environment window. You can now customize the form, and create or modify the form's controls using the instructions in the following sections.

5. Select **File > Save** or click the **Save** button .

The Form Name dialog box appears with the name of the form filled in. If this is the first form for the application, DroidDB automatically gives the form the same name as the application, and you cannot change it.

The first form you create for an application is automatically its primary form. The primary form always has the same name as the application. The primary form is the form that opens when users start your application on the Android device.

Many applications have just one form, but more complex applications may have any number of forms. If the new form is not the primary form, you can assign it any name of your choosing, but typically it is easiest if you simply give the form the same name as the table it is built on.

## Modifying an Existing Form

All changes to an existing form must be made in DroidDB's development environment.

You can make the following types of changes:

- Adding new controls
  - Changing an existing control's properties
  - Deleting and recreating a control
  - Respecifying the form title
  - Defining a new initial sort/search column
1. Working in the DroidDB development environment, open the DroidDB application to which the form belongs.
  2. Select **File > Open** and select the form you wish to modify.
  3. Once you have completed your changes, select **File > Save**.

## **Deleting a Form**

It is very easy to delete a form from an application.

1. Working in DroidDB's development environment on the desktop PC, open the DroidDB application to which the form belongs.
2. Select **File > Delete Form** and select the form you wish to delete.

# Customizing Forms

## Specifying the Form Size

By default, the form size is the same as the device screen size. You can specify a form size that is taller than the Android device's screen size.

### Notes

- The device screen size is indicated by the dotted-line frame in the development window. If this outline does not match the size and shape of your target Android device screen, you can change it using the instructions on page 86.
  - As an alternative to making your form larger, you can divide your form into tabbed pages, as described on page 117.
1. If the form is not already open in the development environment window, select **File > Open**. When DroidDB displays the Form name dialog box, select the desired form.
  2. Select **Form > Form Size**.

DroidDB displays the Form Size dialog box.



3. Enter the height of the form (in pixels), then click **OK**.

When a form that is taller than the device screen size is run on the Android device, users can bring hidden parts of the form into view by swiping the form up and down.

### Tips

- You can verify how much of the form can be displayed on the Android device screen at any one time. To do this, click inside the dotted-line frame while pressing the Shift key, and drag the frame up and down in the development environment window. Any controls that appear within the frame will appear together on the Android device screen.
- After you change the form size, you may want to change the size of the development window so you can see more of the form at one time. Mouse over the lower right corner of the window so that the pointer changes to a double-headed arrow. Click and drag the corner outward.

## Specifying Form Background Color

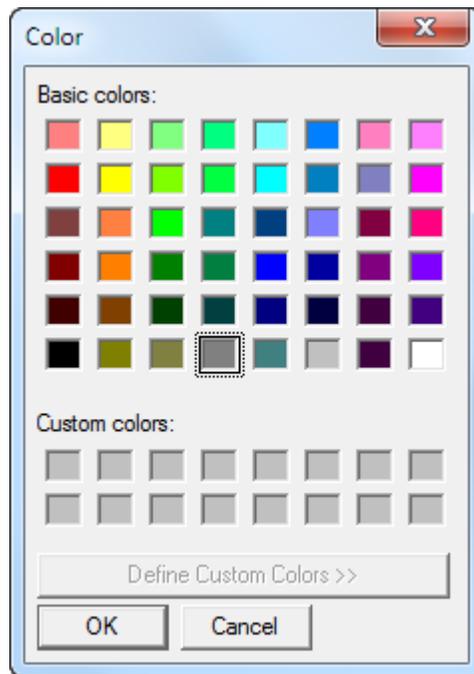
Color can add interest and clarity to your forms.

### Note

You can choose from the basic palette of 48 pre-defined colors, or you can add up to 16 custom colors to the selection. Refer to the topic, "Adding Custom Colors to the Color Palette," that follows.

1. If the form is not already open in the development environment window, select **File > Open**. When DroidDB displays the Form name dialog box, select the desired form.
2. Select **Form > Color**.

DroidDB displays the Color palette.



3. Click one of the color swatches, then click **OK**.

DroidDB displays the form with your selection in the background.

### Tip

As you create controls, you can specify their colors as well (refer to the instructions for the particular control type). You can also apply a color to several controls at once (refer to the topic, "Changing Properties for Multiple Controls," on page 175).

## Adding Custom Colors to the Color Palette

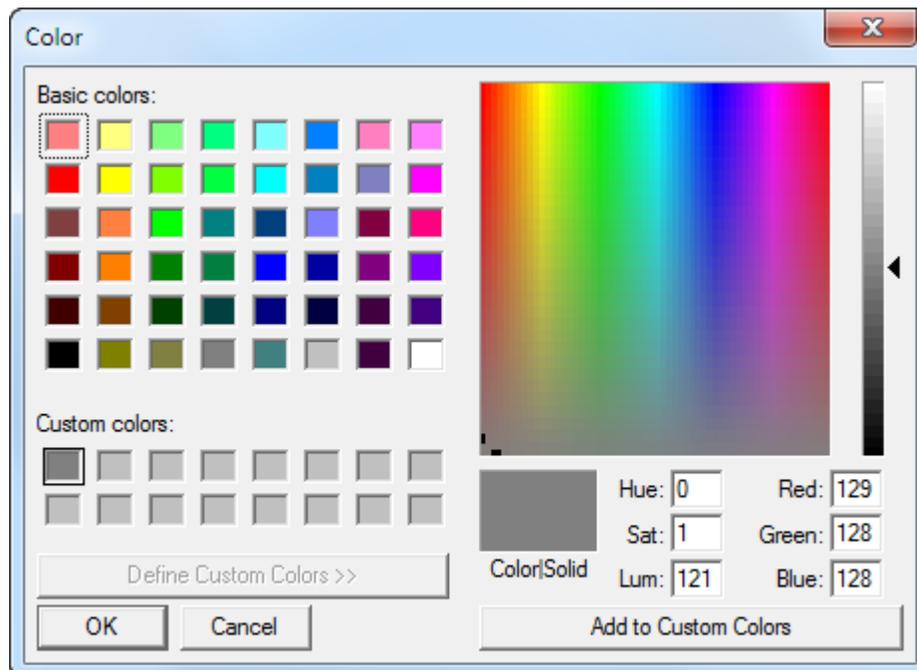
You can add up to 16 custom color swatches to the basic pre-defined palette of 48. The colors are then available to apply to any element of the form (background, controls, fonts, etc.).

You must save the application to save the custom palette. Also, custom color swatches are saved with just the current form.

To add a custom color swatch:

1. Select **Edit > Custom Colors**.

DroidDB displays the Windows Color dialog box.



2. Select one of the empty boxes under "Custom colors" in the lower-left portion of the dialog box. (The box you select is where the new color swatch will be placed.)
3. Define a new color using the controls on the right side of the dialog box, for example:
  - Click anywhere in the color matrix (the cross-hairs identify the area you clicked). The resulting color appears in the "Color/Solid" box.
  - Alternatively, you can define a color numerically by typing values for Hue, Saturation, and Luminosity; or Red, Green, and Blue. You can also change the luminosity (lightness/darkness) using the vertical slider. **Tip:** The RGB values for Android green are 164 198 57.
4. Once you are satisfied with the color as it appears in the "Color/Solid" box, click **Add to Custom Colors**.

The new color swatch appears under "Custom colors."

If you want to replace an existing custom color swatch: click it, redefine the color as described above, and click **Add to Custom Colors**.

5. Click **OK** to save your new color palette.

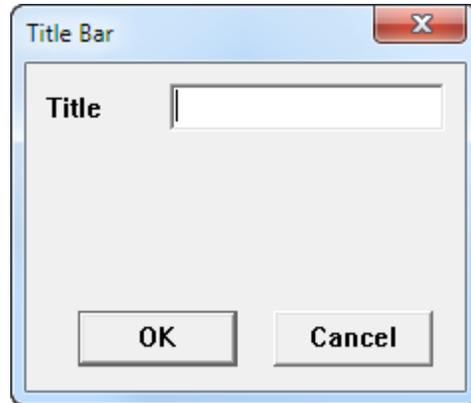
## Adding a Title

You can specify a title for your form. The title displays in the title bar in both the development environment window and the application window.

1. If the form is not already open in the development environment window, select **File > Open**. When DroidDB displays the Form name dialog box, select the desired form.

2. Select **Form > Title Bar**.

DroidDB displays the Title Bar dialog box.



3. Enter the title in the Title field, and click **OK**.

DroidDB displays your title at the top of the development window.

## Adding Tabs

Tabs enable you to group the controls on your form into pages and give your users quick access to each page. If you took Quick Tour I, you are already familiar with tabs.

### Tips

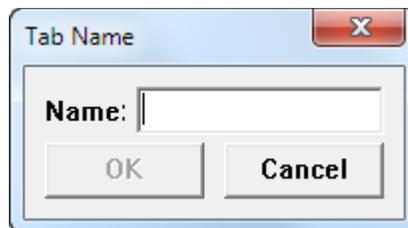
- It's generally easier to create the tabs, then create the controls on each tabbed page. If you create the controls first, and then the tabs, DroidDB automatically puts all of the controls on the first page, in which case you'll have to cut and paste some of the controls onto the other tabbed pages.
  - The tabs are displayed at the top of the application window.
1. If the form is not already open in the development environment window, select **File > Open**. When DroidDB displays the Form name dialog box, select the desired form.
  2. Select **Form > Tabs**.

DroidDB displays the Tabs dialog box.



3. Select **Add Before**.

DroidDB displays the Tab Name dialog box.



4. In the Name field, enter the text you want to appear on the first tab in the application window, and click **OK**.
5. If desired, click the **Text** button to assign a color to the tab text, and click the **Back** button to assign a color to the tab background.

6. Add more tabs:
  - Click an existing tab name in the Tabs window to highlight it.
  - Click the **Add After** button if the new tab is to follow the highlighted tab, or **Add Before** if the new tab is to precede it. (The tabs will appear from left to right across the top of the application window in the order that they appear in the Tabs window.)
  - When the Tab Name dialog opens, enter the tab text in the Name field and click **OK**.
7. If you want to change the order of the tabs as they will appear on your form, click a tab name to highlight it, then click the **Move Up** or **Move Down** button.
8. When you have finished defining the tabs, click **OK** on the Tabs dialog box.

DroidDB displays the tabs in the development window as they will appear on your form.
9. To add controls to each tabbed page, simply select the tab in the development window and add controls as you normally would.

## Customizing or Removing the Menu Bar

The standard DroidDB application window contains a menu bar with two menus (Record and Option). You can replace this default configuration with up to four custom menus. This feature makes it easy to provide menus precisely tailored to user needs and to create applications for non-English speaking users.

When creating a custom menu, you can include any of the standard DroidDB commands named as desired, as well as commands to launch macros that you have created.

You also have the option of removing the menus and toolbar buttons from the application window altogether. In that case, you can give users access to a customized set of commands by adding command buttons to your form. Refer to the topic, "Creating Command Buttons," on page 154.

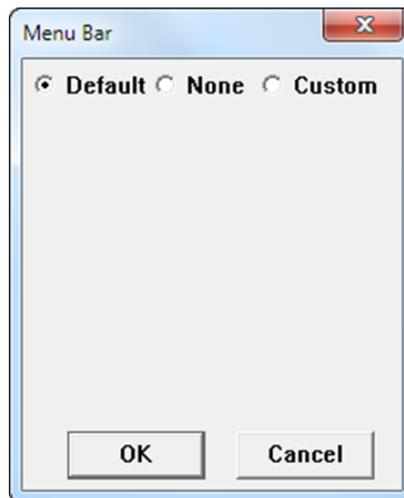
### Tip

- If you hide the menu bar, users will have no obvious way to close the form. You might consider supplying a command button for that purpose (Record | Close).

### To customize or remove the menu bar:

1. Select **Form > Menu Bar**.

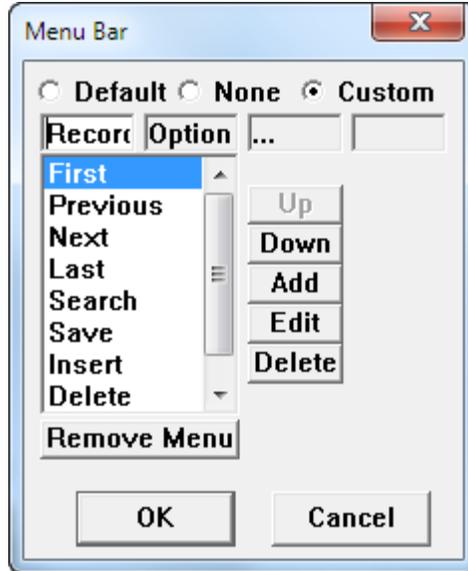
DroidDB displays the Menu Bar dialog box.



2. Choose one of three options:

- **Default:** Display the standard menus (Record and Option).
- **None:** Remove the menu bar from the application window.
- **Custom:** Create your own menus for the application window. You can simply add a new menu to the default set; modify or delete the default menus; and/or create all new menus.

If you select "Custom", the dialog box displays additional options. Proceed with Step 3. Otherwise, click OK.



3. You can have one to four menus. You can add new menus, revise or reuse existing menus, or delete menus. Detailed instructions are provided below.

**To add a new menu:**

1. Click in the box containing the three ellipses (...), and enter a name for the new menu.

**Note**

Each new menu you create is automatically added to the right of existing menus. If you want a new menu to appear to the left of an existing menu (for example, Option), you would have to first delete the Option menu, add the new menu, then recreate the Option menu. If you are creating all new menus, be sure to add them in the order you wish them to appear from left to right.

2. You can add two types of items to a menu: standard DroidDB commands named as desired, and commands to run macros you've previously created. The method to add each type is described below:
  - **To add a DroidDB command:** Click the **Add** button. Click the small down arrow to the right of the Action field, and select the desired command from the list. If desired, enter a new name to identify the command in the menu. Click **OK**.
  - **To add a command to run a macro:** *To use this feature, you must have previously created a macro.* Click the **Add** button. Click the small down arrow to the right of the Action field, scroll to the bottom of the list, and select **Run Macro**. In the Macro field, select the macro to run. In the Name field, (optionally) enter a different name to appear in the menu. Click **OK**.
3. Repeat step 2 for each item you wish to include in the menu.

If desired, you can modify the menu using the instructions below.

**To modify a menu (reorder, rename, add, or delete menu commands):**

1. In the Menu Bar dialog box, click in the box containing the name of the menu to be modified (e.g., Option).

The commands that currently make up the menu appear below its name.

2. Modify the menu as desired:
  - To reorder the menu items, click an item to highlight it, then click either the **Up** or **Down** button to change the item's position in the list.
  - To rename a command, click the command to highlight it, then click the **Edit** button. Enter the new name in the Name field and click OK.
  - To associate a different action with a command, click the command to highlight it, then click **Edit**. Click the small down arrow to the right of the Action field, and select the desired action. Click OK.
  - To add a command or macro, click the **Add** button and proceed as described in Step 2 in the section above, "To add a new menu."
  - To remove a menu item, click the item to highlight it, then click the **Delete** button.
3. When you have finished modifying the menu, click **OK**.

**To remove a menu:**

1. In the Menu Bar dialog box, click in the box containing the name of the menu you wish to remove (for example, Record).
2. Click the **Remove Menu** button.

DroidDB asks you to confirm that you want to remove the menu. If you respond "Yes," DroidDB removes the entire menu and slides the remaining menus to the left.

# Adding Controls to the Form

## About Controls

You can create any of the following types of controls in your DroidDB application:

- Edit boxes – see page 123
- Note boxes – see page 129
- Labels – see page 131
- Checkboxes – see page 132
- Radio buttons – see page 133
- Drop-down lists – see page 135
- Calculated fields – see page 151
- Command buttons – see page 154
- Scribble boxes – see page 158
- Image controls – see page 159 and 160
- Jump buttons – see page 161
- Lookups – see page 165
- Grid controls – see page 168

## Creating Edit Boxes

An edit box is a single-line text box. It can display the value of a table column or global variable, and/or accept the user's entry of a value for a table column or global variable.

Edit boxes can be read-only or read/write. An edit box can be used to display data from any type of column in the table, except Yes/No. An edit box can be associated with a global variable that stores text, integers, floating point numbers, or date/times. DroidDB displays any edit box value as a character string.

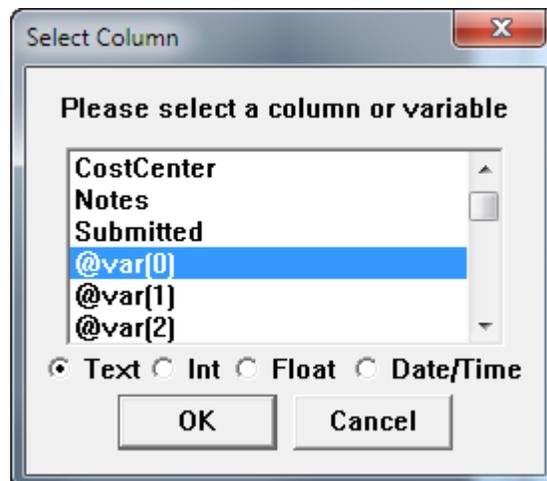
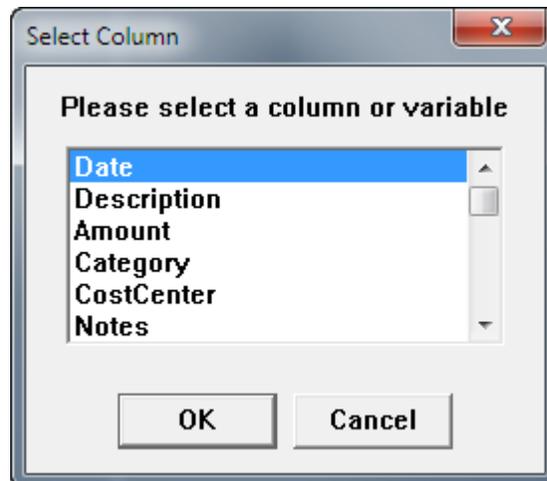
If an edit box is associated with a column or variable that stores integers, floating point numbers, or date/time values, and the user leaves the field blank, DroidDB stores a null value. Alternatively, you can specify a default value that fills the field automatically.

To create an edit box:

1. Select **Control > Edit**, or click the **Edit** button , or right-click where you want the control to appear and select **Edit**.

DroidDB displays the Select Column dialog box.

2. Select the table column or global variable to which the edit box relates. If you select a variable, you must also choose the datatype that the control will read/write to the variable.



3. Click **OK**.

DroidDB displays the Edit Properties dialog box, which contains the following fields:

Field	Description
Connected to [column or variable]	The table column or global variable to which the control relates – cannot be changed.
Left	The initial horizontal position of the left side of the control.
Top	The initial vertical position of the top of the control.
Width	The initial width of the control.
Height	The initial height of the control.
Font	The font, size and style of the type that appears in the edit box. Click to modify.  Refer to the topic, "Specifying Font Properties for a Control," on page 174 for instructions.
Events	Optional. Select macros to be triggered when the user puts the focus on and/or moves the focus away from the control.  Refer to "Triggering Macros with Events," on page 227.
Text	The color of the type that appears in the edit box.
Back	The color of the background in the edit box.
Read-only	If checked, users cannot change the control's value.
Password	<b>Text Boxes Only.</b> If checked, characters entered by the user are hidden by bullets.
Default	The default value for the field. (When a new record is created, the edit box initially contains the default value automatically.)  Refer to the topic, "Edit Box Default Values," on page 125 for details.
Money	If the column or variable you specified is associated with floating point numbers, the Money checkbox displays. If Money is checked, entries are displayed with two decimal places.  Refer to the topic, "Edit Box Money Values," on page 127 for details.
[Date/time options]	If the column or variable is associated with dates/times, four options are available for the type of data that can be entered and displayed: Long Date, Short Date, Time, or Date/Time. Note: The display format associated with each option is determined by the Date/Time Settings on the device running the form.  Refer to the topic, "Edit Box Date/Time Formats," on page 128 for details.

4. Make any changes required to the edit box properties, then click **OK**.

5. Drag the edit box to the correct location.

6. Create a label to identify the edit box.

## Edit Box Default Values

The options available in the Default field of the Edit Properties dialog box depend upon whether the table column or global variable associated with the control is defined as text, numeric, or date/time.

### Text Edit Controls

For text edit controls, enter the default text, if any.

### Numeric Edit Controls (Integer or Floating Point Number)

The initial default is 0. Alternatively, you can specify that the default be no value (null) or a particular value.

*To specify that there is no default value, click the **Null** checkbox. If the user does not supply a value in the edit box, DroidDB stores a null value in the table column or global variable.*

*To specify a particular value as the default:*

1. If the column is a money column, click the **Money** checkbox first. (If it is not a money column, go to Step 2.)

DroidDB inserts the decimal point in your default entry.

2. Click the Default button .
3. DroidDB displays a number pad dialog box. (The number pad includes a decimal point key for floating point numbers.)



4. Click the buttons to enter the default amount – the number you enter displays in the title bar of the dialog box (if you make a mistake, click the Clear button, then enter the number again).
5. To confirm your entries, click **OK**.

### Date/Time Edit Controls

The initial default is Now (the current date and/or time). Alternatively, you can specify that the default be no date/time (null) or a particular date and/or time.

*To specify that there is no default date/time, click the **Null** checkbox.*

To specify a particular date and/or time as the default:

1. Select an option for the amount of information stored in the date/time column:

Option	Sample
Long Date	Friday, September 30, 2012
Short Date	9/30/2012
Time	2:28:00 PM
Date/Time	9/30/2012 2:28:00 PM

**Note**

The Date and Time Settings on the Android device determine the format that DroidDB uses to display the selected option. Refer to the topic, "Edit Box Date/Time Formats," on page 128 for details.

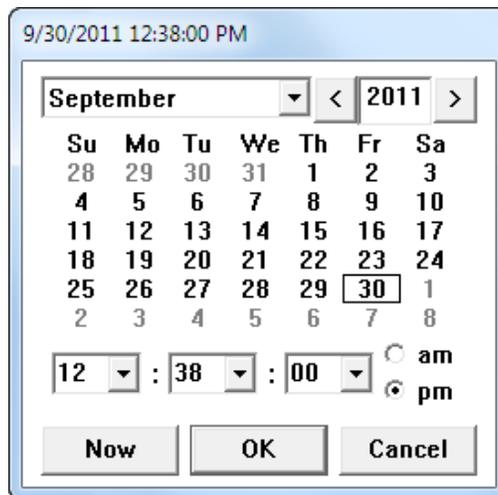
2. Click the **Now** or **Null** checkbox so that neither box is checked.

DroidDB displays a button that identifies the date and/or time to be used as the default in the application.



3. Click the date/time button.

DroidDB displays a calendar dialog box. The elements displayed correspond to the format you specified in Step 1. For example, if you specified Time, only the time selectors are available.



4. Specify the default date and/or time (again, only the options that correspond to the option you selected in Step 1 are available):
  - Click the month (from the drop-down list)
  - To change the year, enter it directly, or click the right-/left-arrow to increase/decrease the year
  - Click the day in the calendar area
  - To change the time, type hours and minutes directly or select from the drop-down lists; choose either am or pm
  - To change the default back to the current date/time, click **Now**
  - To confirm your entries, click **OK**

## Edit Box Money Values

For columns or variables that contain floating point numbers, the Edit Properties dialog box displays the Money checkbox. The Money checkbox is not checked initially.

To identify numeric entries as money values:

1. Click the **Money** checkbox to check it.

DroidDB updates the Default button to 0.00 (showing two decimal places).



2. To enter a specific money value as the default:
  - Click the **Default** button (with the 0.00 label)  
DroidDB displays a number pad dialog box, which includes a decimal point key for money entries.
  - Click the number buttons to enter the default amount – the number you enter displays in the title bar of the dialog box  
(If you make a mistake, click the Clear button, then enter the number again.)
  - To confirm your entry, click **OK**

## Edit Box Date/Time Formats

For date/time columns or variables, the Edit Properties dialog box enables you to choose between four options for the amount of information stored: Long Date, Short Date, Time, and Date/Time.

<u>Option</u>	<u>Sample</u>
Long Date	Friday, September 30, 2012
Short Date	9/30/2012
Time	2:28:00 PM
Date/Time	9/30/2012 2:28:00 PM

DroidDB uses the display format specified in the Android device's Date and Time settings. For example, possible display formats for long dates include Monday, April 26, 2012 and 26 April, 2012; short dates include 4/26/12 and 2012-04-26.

It's recommended that you match the date/time format settings on the desktop PC to those on the Android device when you create the form, so that you can size the controls to accommodate the length of the dates/times to be displayed.

## Creating Note Boxes

A note box is a multi-line text box. A note box can display the value of a table column or global variable, and/or accept the user's entry of a value for a table column or global variable.

Note boxes can be read-only or read/write. A note box can be used to display any column in the table, except Yes/No. A note box can be associated with a global variable that stores text, integers, floating point numbers, or date/times.

DroidDB displays any note box value as a character string.

To create a note box:

1. Select **Control > Note**, or click the **Note** button , or right-click where you want the control to appear and select **Note**.

DroidDB displays the Select Column dialog box.

2. Select the table column or global variable to which the note box relates. If you select a variable, you must also choose the datatype that the control will read/write to the variable.
3. Click **OK**.

DroidDB displays the Note Properties dialog box, which contains the following fields:

Field	Description
Connected to [column or variable]	The table column or global variable to which the control relates – cannot be changed.
Left	The initial horizontal position of the left side of the control.
Top	The initial vertical position of the top of the control.
Width	The initial width of the control.
Height	The initial height of the control.
Font	The font, size and style of the type that appears in the edit box. Click to modify. Refer to the topic, "Specifying Font Properties for a Control," on page 174 for instructions.
Events	Optional. Select macros to be triggered when the user puts the focus on and/or moves the focus away from the control. Refer to "Triggering Macros with Events," on page 227.
Text	The color of the type that appears in the edit box.
Back	The color of the background in the edit box.
Read-only	If checked, users cannot change the control's value.
Default	The default value for the field. Refer to the topic, "Edit Box Default Values," on page 125 for details.
Money	If the column or variable you specified is associated with floating point numbers, the Money checkbox displays. If Money is checked, entries are displayed with two decimal places. Refer to the topic, "Edit Box Money Values," on page 127 for details.

[Date/time options] If the column or variable is associated with dates/times, four options are available for the type of data that can be entered and displayed: Long Date, Short Date, Time, or Date/Time. Note: The display format associated with each option is determined by the Date/Time Settings on the device running the form.

Refer to the topic, "Edit Box Date/Time Formats," on page 128 for details.

4. Make any changes required to the edit box properties, then click **OK**.
5. Drag the note box to the correct location.
6. Create a label to identify the note box.

## Creating Labels

Labels display text that the user cannot modify. A label can be used to identify other controls, such as edit boxes and note boxes, that do not have their own captions. DroidDB always displays the same value, regardless of the current record.

Labels are always read-only. Unlike other DroidDB controls, labels are not associated with table columns or global variables.

To create a label:

1. Select **Control > Label**, or click the **Label** button , or right-click where you want the control to appear and select **Label**.

DroidDB displays the Label Properties dialog box, which contains the following fields:

Field	Description
Text	Enter the text of the label. You can use spaces and carriage returns to format the text.
Left	The initial horizontal position of the left side of the control.
Top	The initial vertical position of the top of the control.
Width	The initial width of the control.
Height	The initial height of the control.
Font	The font, size and style of the type for the label. Click to modify. Refer to the topic, "Specifying Font Properties for a Control," on page 174 for instructions.
Text	The color of the label's type.
Back	The color of the label's background.
Left / Center / Right	Alignment of text within the box that contains it: left aligned, centered, or right aligned.

2. Make any changes required to the label properties, then click **OK**.
3. Drag the label to the correct location.
4. In necessary, change the label's width and/or height.

## Creating Checkboxes

A checkbox indicates a choice or set of choices from which the user can select none, one, or more than one option.

Checkboxes can be read-only or read/write. Checkboxes can be associated with any text, integer, floating point number, memo, or yes/no column in the table; they cannot be associated with date/time columns. A checkbox can be associated with a global variable.

If the checkbox is associated with text or memos, DroidDB checks the box for strings whose first value is either 1, Y, y, T, or t, and clears the box for strings that do not begin with any of those values.

If the checkbox is associated with an integer or floating point number, DroidDB checks the box for any non-zero value, and clears the box for any zero value.

To create a checkbox:

1. Select **Control > Checkbox**, or click the **Checkbox** button , or right-click where you want the control to appear and select **Checkbox**.

DroidDB displays the Select Column dialog box.

2. Select the table column or global variable to which the checkbox relates, then click **OK**.

DroidDB displays the Checkbox Properties dialog box, which contains the following fields:

Field	Description
Connected to [column or variable]	The table column or global variable to which the control relates – cannot be changed.
Left	The initial horizontal position of the left side of the control.
Top	The initial vertical position of the top of the control.
Width	The initial width of the control.
Height	The initial height of the control.
Font	The font, size and style of the type that appears in the edit box. Click to modify. Refer to the topic, "Specifying Font Properties for a Control," on page 174 for instructions.
Events	Optional. Select macros to be triggered when the user puts the focus on and/or moves the focus away from the control, or changes the value in the control. See "Triggering Macros with Events," on page 227.
Text	The color of the caption displayed to the right of the checkbox.
Back	The color behind the checkbox and caption.
Read-only	If checked, users cannot change the control's value.
On by default	If checked, the checkbox is initially checked when a new record is created. If not checked, the checkbox is initially clear when a new record is created.
Text	The caption to be displayed to the right of the checkbox.

3. Make any changes required to checkbox properties, then click **OK**.
4. Drag the checkbox to the correct position, then resize it if necessary.

## Creating Radio Buttons

Radio buttons offer a set of choices from which the user can select one option. You can require the user to explicitly select an option, or you can provide a default value.

Radio buttons can be read-only or read/write. They can be associated with any text, integer, floating point number, or memo column in the table; they cannot be associated with date/time or yes/no columns. Radio buttons can be associated with global variables that store text, integers, or floating point numbers.

If radio buttons are associated with text or memos, the caption of the button selected is the value stored in the table or variable.

If the radio buttons are associated with an integer or floating point number, you must specify a caption and a unique numeric code for each option. While the user sees the caption, it is the number code that is stored.

DroidDB creates a radio button for each caption you specify. You can specify up to eight radio buttons per set of radio buttons.

You can arrange the radio buttons in a vertical stack or horizontal line.

To create radio buttons:

1. Select **Control > Radio buttons**, or click the **Radio Buttons** button , or right-click where you want the control to appear and select **Radio Buttons**.

DroidDB displays the Select Column dialog box.

2. Select the table column or global variable to which the radio buttons relate. If you select a variable, you must also choose the datatype that the control will read/write to the variable. Click **OK**.

DroidDB displays the Radio Buttons Properties dialog box, which contains the following fields:

Field	Description
Connected to [column or variable]	The table column or global variable to which the control relates – cannot be changed.
Left	The initial horizontal position of the left side of the control.
Top	The initial vertical position of the top of the control.
Width	The initial width of the control.
Height	The initial height of the control.
Font	The font, size and style of the type that appears in the edit box. Click to modify.  Refer to the topic, "Specifying Font Properties for a Control," on page 174 for instructions.
Events	Optional. Select macros to be triggered when the user puts the focus on and/or moves the focus away from the control, or changes the value in the control.  Refer to "Triggering Macros with Events," on page 227.
Text	The color of the radio buttons' captions.

Back	The color behind the radio buttons and captions.
Read-only	If checked, users cannot change the control's value.
Horizontal	If checked, the radio buttons are arranged in a row on the form. If unchecked, they are stacked one above the other.

- After making any changes necessary to radio button properties, click the **Edit** button to specify the properties of the first button in the group.

DroidDB displays the Option Specification dialog box, which contains the following fields:

Field	Description
Name	The caption to be displayed to the right of the radio button.
Default	If checked, this radio button is initially selected when a new record is created. Only one radio button can be the default.
Code	Only shown if the control is associated with a column or variable that stores integers or floating point numbers. Enter a unique numeric code for each radio button specified.

- Make the entries required for the first radio button caption, then click **OK**.

**Note**

DroidDB displays an asterisk (\*) after the caption of the default radio button.

- Specify the remaining buttons in the group:
  - Click an existing radio button caption to highlight it.
  - Click the **Add After** button if the new caption is to follow the highlighted caption, or **Add Before** if the new caption is to precede the highlighted caption.
  - Enter the new caption in the Name field, (click Default if applicable).
  - If you want to change the order of the radio buttons as they will appear on the form, click a radio button caption to highlight it, then click the **Move Up** or **Move Down** button.
  - When you have finished specifying the properties, click **OK**.
- If necessary, drag the control to the correct location, and resize it so all the radio buttons and their names can be clearly read.

## Creating Drop-Down Lists

A drop-down is a scrollable list of choices from which the user can choose one, or (for text or memos) enter a new option. Drop-down lists are especially useful for enabling users to pick from a group of frequently used values without having to type, or for restricting input to a set of allowable values. You can provide a default value. If you don't explicitly define a default, the first item in the list becomes the default.

Drop-downs are similar to radio buttons, but they offer additional advantages: 1) The list of options can be any length (not restricted to the space available on the form); 2) You can allow users to type in new options.

Drop-downs can be associated with any text, number, or memo column in the table; they cannot be associated with date/time or yes/no columns. They can be associated with global variables that store text, integers, or floating point numbers.

If the drop-down list is associated with text or memos, the name that the user picks from the drop-down list is the value stored in the table or variable. If the drop-down list is associated with a numeric column or variable, you must specify a name and a unique numeric code for each option. While the user always sees the name, it is the number code that is stored.

**User-defined entries:** For text and memos, you can allow users to type new values (rather than just pick from the list). You cannot allow user-defined entries for numeric columns or variables.

To create a drop-down list:

1. Select **Control > Drop Down**, or click the **Drop Down** button  , or right-click where you want the control to appear and select **Drop Down**.

DroidDB displays the Select Columns dialog box.

2. Select the table column or global variable to which the drop-down list relates. If you select a variable, you must also choose the datatype that the control will read/write to the variable. Click **OK**.

DroidDB displays the Drop Down Properties dialog box, which contains the following fields:

Field	Description
Connected to [column or variable]	The table column or global variable to which the control relates – cannot be changed.
Left	The initial horizontal position of the left side of the control.
Top	The initial vertical position of the top of the control.
Width	The initial width of the control.
Height	The number of entries displayed at one time in the scrollable list box.
Font	The font, size and style of the type that appears in the list. Click to modify. Refer to the topic, "Specifying Font Properties for a Control," on page 174 for instructions.
Events	Optional. Select macros to be triggered when the user puts the focus on and/or moves the focus away from the control, or changes the value in the control.  Refer to "Triggering Macros with Events," on page 227.

Text	The color of the type that appears in the list.
Back	The color of the background inside the drop-down list box.
User can add	(Available only for text or memos.) If checked, users can type new values, rather than just pick from the list.
Include values from table	Check to specify that DroidDB create the list from the contents of a column in another table. (Refer to “Using Values from Another Table as a Drop-Down List” on page 137 for details.)

3. After making any changes necessary to drop-down list properties, click the **Add Before** button.

DroidDB displays the Option Specification dialog box, which contains the following fields:

Field	Description
Name	The text to be displayed in the drop-down list for this option.
Default	If checked, this option is initially selected when a new record is created. Only one option in the drop-down list can be the default.
Code	If the control is associated with an integer or floating point number column or variable, this is the numeric code for the option.

4. Make the entries required for the first drop-down list caption, then click **OK**.

**Note**

DroidDB displays an asterisk (\*) after the list's default option.

5. Specify the remaining drop-down list options:

- Click an existing caption to highlight it.
- Click the Add After button if the new caption is to follow the highlighted caption, or Add Before if the new caption is to precede the highlighted caption.
- Enter the option's caption in the Name field.
- Click Default if this option is to be the default.
- If the column or variable is numeric, enter a unique value for the option in the Code field.
- Click OK.

**Tip**

If you want to change the order of the drop down list options as they will appear on the form, click a caption to highlight it, then click the **Move Up** or **Move Down** button.

6. Click **OK** on the Drop Down Properties dialog box.
7. Drag the drop-down list to the correct location, and resize it if necessary.

## Using Values from Another Table as a Drop-Down List

You can make it possible for users to complete a field by selecting from items stored in a table. More specifically, you can specify a drop-down list that DroidDB dynamically composes from the current contents of a column in any table belonging to the application.

To create a drop-down list using values from a table:

1. Select **Control > Drop Down**, or click the **Drop Down** button , or right-click where you want the control to appear and select **Drop Down**.

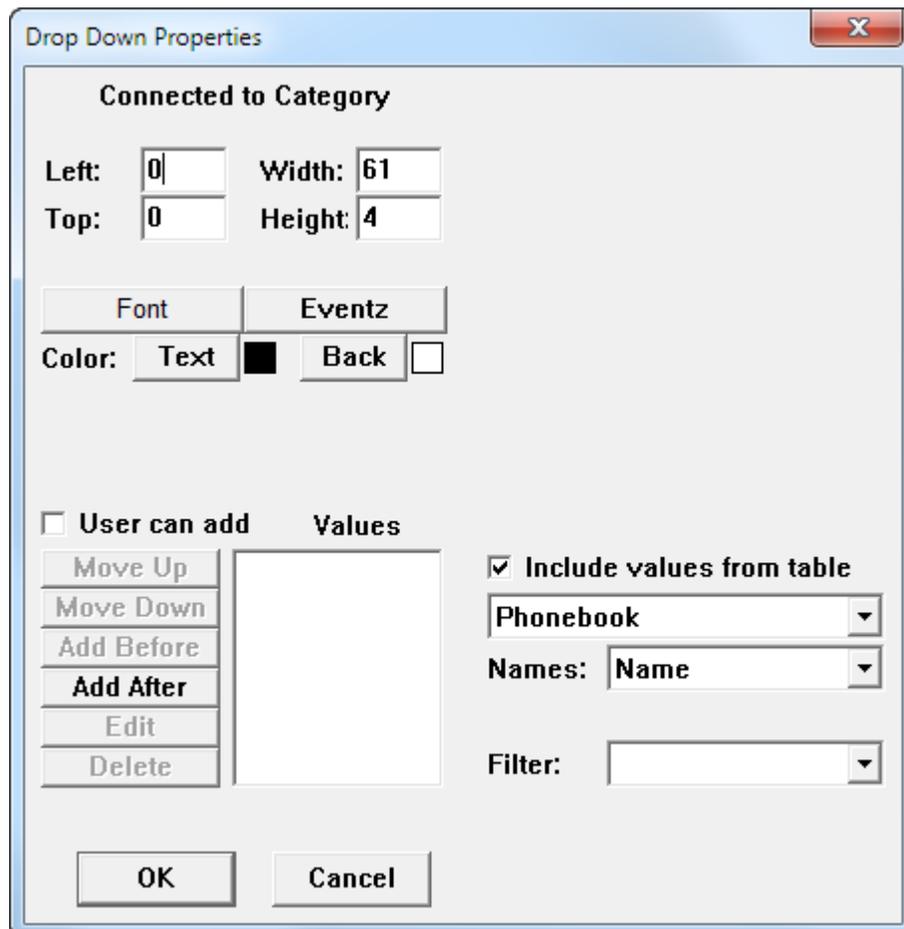
DroidDB displays the Select Columns dialog box.

2. Select the table column or global variable to which the drop-down list relates. If you select a variable, you must also choose the datatype that the control will read/write to the variable. Click **OK**.

DroidDB displays the Drop Down Properties dialog box.

3. In the Drop-Down Properties dialog, select the **Include values from table** checkbox. (Complete the other options as you normally would for drop-down lists. See the topic, "Creating Drop-Down Lists," on page 135.)

DroidDB searches the connected Android device to determine eligible tables. DroidDB then displays additional fields in the lower right corner of the Drop Down Properties dialog.



**Drop Down Properties**

**Connected to Category**

Left:  Width:   
 Top:  Height:

Font      Eventz

Color: Text  Back

**User can add Values**

**Include values from table**

Phonebook

Names: Name

Filter:

OK      Cancel

4. Complete the additional fields:
  - In the first new field (directly under "Include values from table"), select the table that contains the values you want to use for the drop-down list.
  - In the **Names** field, select the column that contains the text you want to appear as options in the drop-down list.
  - If you selected a numeric column or variable in Step 2, DroidDB displays a third field, **Codes**. Select the column that contains the associated code values to be stored instead of the names.
  - **Filter:** Leave blank unless you are creating a dependent drop-down list. Refer to the topic, "Creating Dependent Drop-Down Lists," on page139 for details.
5. Click **OK** on the Drop Down Properties dialog box.
6. Drag the drop-down list to the correct location, and resize it if necessary.

## Creating Dependent Drop-Down Lists

If your application contains a field whose set of allowable values is dependent upon a value in another field, you may want to incorporate a "dependent drop-down list" in your design.

The options in a dependent drop-down list are automatically "filtered" according to the value that the user selects in another control on the form (call this value the "filter").

Consider the following example. Say your table has two columns: Feature and Color. Not all colors are appropriate choices for all features—for example, blue is a typical color for eyes but not for hair. You could set up your form so that the names displayed in the Color drop-down list depend on the user's selection for Feature. Feature is the filter, and Color is the dependent. If the user selects the feature Hair, the dependent drop-down list would offer the choices Black, Blond, Brown, and Red. But if the user selects Eyes, the dependent drop-down list would offer Blue, Brown, and Green.

All of the rules and instructions described in the previous topic, "Using Values from Another Table as a Drop-down List," on page 137 apply to dependent drop-downs.

The control for selecting the filter value in the main form can be a drop-down list, but it doesn't have to be. You can associate the filter control with a table column if you want to store the user's entry for the filter value, or you can associate it with a global variable if you don't.

### Prerequisite: Supporting Table to Store the Drop-Down Options

To incorporate dependent drop-downs in your new application, you must create a separate table in the application and fill it with the filter and dependent drop-down options.

The support table and its contents must meet the following requirements:

- There must be at least two columns in the support table: one to store the filter values and one to store the corresponding dependent option Names.

#### Note

Optionally, you can utilize space-saving numeric codes for the dependent and/or filter values, just as you would for any other drop-down list control. For example, you could create two columns for the dependent values: a text column to store the names that appear on the form in the list display, and a numeric column to store the invisible codes associated with each name. The user sees the name (e.g., black), but it is the more compact numeric code that is stored in the main table (e.g., 1).

- If you intend to store the user's selection for the filter value, the main table must have a column with the same name (e.g., Feature) and datatype (e.g. text) as the column that stores filter values in the support table. Alternatively, if you don't want to store the user's selection for the filter value, you can associate the filter control in the main form with a global variable. In that case, the variable's datatype must match that of the column in the support table.
- The column that stores the filter values in the support table must be indexed.
- Once you've created the support table, you must enter one unique row (record) for each filter/dependent pair. So for example, if you have two filter options--one with four dependent options and the other with three--you would have to create a total of seven records:  $(1 \times 4) + (1 \times 3) = 7$ .

The supporting table for the Feature/Color example might be organized like this:

<b>Feature*</b> <b>(Filter)</b>	<b>Color</b> <b>(Dependent Name)</b>	<b>ColorID**</b> <b>(Dependent Code)</b>
Hair	Black	1
Hair	Blond	2
Hair	Brown	3
Hair	Red	4
Eyes	Blue	5
Eyes	Brown	6
Eyes	Green	7
*Indexed		**Optional

There are one or two options for making it possible to add and maintain data in a support table:

- You can build a form on top of the support table, and add a “Run Form” command button on the primary form to open it. (The primary form is the one that opens automatically when the user launches the application. It has the same name as the application.)
- Business Edition only: You can maintain a version of the support table on the desktop PC and update its Android device counterpart via synchronization.

Once you have created and filled the support table and created the corresponding columns in the application table, you are ready to create the filter and dependent drop-down list controls in your form:

1. First, create a control to allow the user to select the filter value (in our example, Feature). Usually this would be a drop-down control, but it does not have to be. If it is a drop-down control, use the same steps you would for any drop-down that utilizes values from another table:

- Select **Control > Drop Down**, or click the **Drop Down** button , or right-click where you want the control to appear and select **Drop Down**.
- In the Select Columns dialog, select the application table column or global variable that will store the selected filter value (in our example, Feature). If you select a variable, you must also choose the datatype that the control will read/write to the variable. Click **OK**.
- In the Drop-Down Properties dialog, select the **Include values from table** checkbox. (Complete the other options as you normally would for drop-down lists. See the topic, "Creating Drop-Down Lists," on page 135.)  
DroidDB displays additional fields in the lower right corner at the bottom of the Drop Down Properties dialog.
- In the first new field (directly under "Include values from table"), select the supporting table that stores the drop-down options.
- In the **Names** field, select the column that stores the text you want to appear as options in this drop-down list (in our example, Feature).
- **Filter:** Leave blank.
- Click **OK** to close the Drop Down Properties dialog box.
- Drag the drop-down list to the correct location, and resize it if necessary. Create a label to identify it.

2. Now, create a second control—this one is the dependent drop-down list control that will enable the user to select the dependent value. (Remember, the contents of this drop-down list will be filtered by the user's selection in the control you created in Step 1). To create the dependent drop-down:
  - Select **Control > Drop Down**, or click the **Drop Down** button  , or right-click where you want the control to appear and select **Drop Down**
  - In the Select Columns dialog, select the main table column or global variable that will store the user's selection for this control (in our example, Color). If you select a variable, you must also choose the datatype that the control will read/write to the variable. Click **OK**.
  - In the Drop-Down Properties dialog, select the **Include values from table** checkbox. (Complete the other options as you normally would for drop-down lists. See the topic, "Creating Drop-Down Lists," on page 135.)  
After a delay of several moments, DroidDB displays additional fields in the lower right corner at the bottom of the Drop Down Properties dialog.
  - In the first new field (directly under "Include values from table"), select the supporting table that stores the drop-down options.
  - In the **Names** field, select the column that stores the text you want to appear as options in the dependent drop-down list (in our example, Color).
  - In the **Filter** field, select the table column or global variable that stores the filter values--the same table column name or global variable that you selected in Step 1. If you pick a variable, DroidDB asks you to specify the Key; that is, the column that stores the filter values in the support table (in our example, Feature).
  - Click **OK** to close the Drop Down Properties dialog box.
  - DroidDB displays a message asking if you want Auto Recalc turned on. Click **Yes**. Auto Recalc insures that the options displayed in the form for the dependent drop-down correspond to the user's selection in the control used to set the filter value. If this feature is turned off, and a user makes first one selection then another for the filter value, the dependent options won't adjust to the revised selection. (You would typically turn Auto Recalc off only if you have calculated fields in your form that you want to update on command, or if your Android device has performance issues. Refer to the topic, "Enabling the Auto Recalc Feature," on page 187 for more information.)
  - Drag the drop-down list to the correct location, and resize it if necessary. Create a label to identify it.
3. Be sure to test your application to confirm that the list of options in the dependent drop-down appear as expected. If not, confirm that the support table meets the prerequisites defined at the beginning of this topic.

**Tip**

You can create a series of dependent drop-downs, in which the options presented in each list are determined by the combination of the user's preceding selections, much like a decision tree. Refer to the topic, "Multi-Tiered Dependent Drop-Down Lists," on page 142 for details.

## Multi-Tiered Dependent Drop-Down Lists

In the previous topic, you learned how to create a drop-down that automatically displays a list of valid options given the user's selection in another control. You can take this a step further by creating a series of dependent drop-downs in which the options available in each succeeding control are determined by the accumulation of the user's preceding selections in other controls. You can think of the sequence of options as a decision tree—each selection leads the user down a different branch. You can have any number of dependent drop-downs in your design.

Multi-tiered dependent drop-downs streamline data entry and help ensure that users make valid selections.

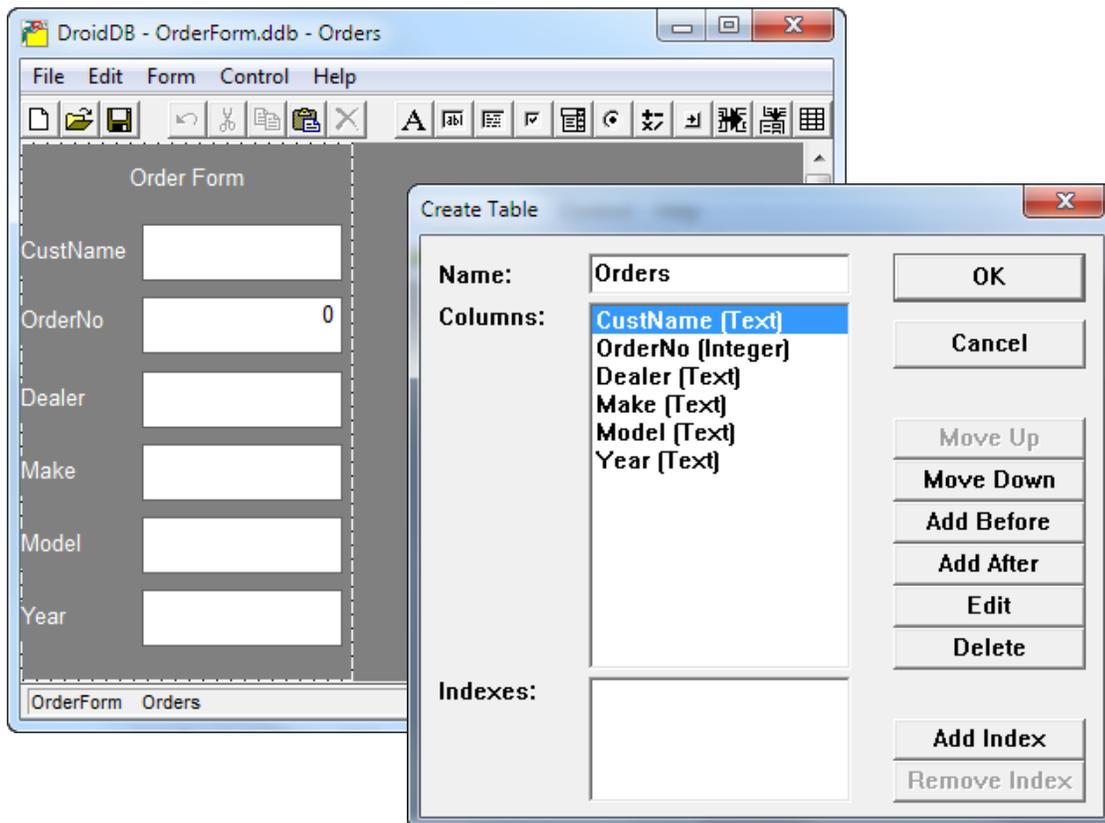
Successful implementation of multi-tiered dependent drop-downs requires careful planning and design. To illustrate this, we will step through the process of building a sample application.

### Note

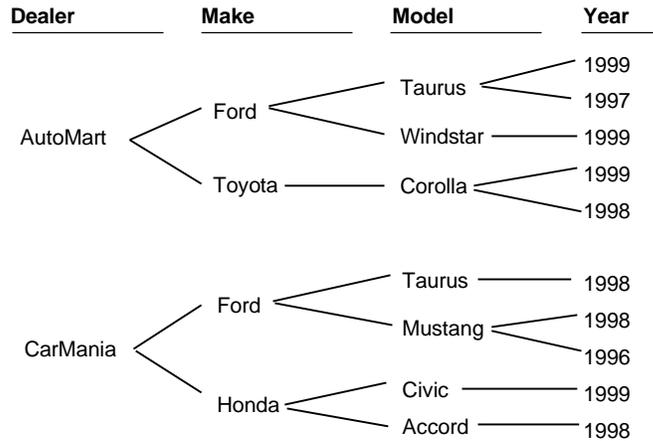
The rules for creating multi-tiered dependent drop-down are the same as those defined in the previous topic, "Creating Dependent Drop-Down Lists," with one important addition: The filter value for each dependent drop-down must uniquely identify the combination of selections that preceded it. We will explain this concept fully in the following example.

### The Scenario

Say that you are a user-car salesperson and want to build an order form that enables you to pick a car by Dealer, Make, Model, and Year. Your initial application might look something like this:



Your form would be much more valuable if it allowed you to pick only cars that are actually in the dealers' inventories. In fact, you know that the range of options is as follows:



You decide to modify your form so that the controls for Dealer, Make, Model, and Year are intelligent drop-downs that display only the options available.

**Design the Support Table**

You know from the previous topic that your first step in creating dependent drop-downs is to create a support table that stores one row for each possible combination of options. Let's call the sample support table, "Inventory." Before you build the table on the computer, it's best to sketch it out on paper. Your initial plan might look like this. Note that every possible branch in the tree is captured.

*Initial Inventory Table Worksheet*

Dealer	Make	Model	Year
AutoMart	Ford	Taurus	1999
AutoMart	Ford	Taurus	1997
AutoMart	Ford	Windstar	1999
AutoMart	Toyota	Corolla	1999
AutoMart	Toyota	Corolla	1998
CarMania	Ford	Taurus	1998
CarMania	Ford	Mustang	1998
CarMania	Ford	Mustang	1996
CarMania	Honda	Civic	1999
CarMania	Honda	Accord	1998

You're not done, though. You need to add another level of information in order for the program to accurately filter the options beyond the Make of the car.

**In order for multi-tier drop-downs to work, the options in each drop-down must be filtered by a value that indicates the combination of the user's selections in the previous drop-downs.** For example, Make is filtered by Dealer; Model is filtered by Dealer and Make; and Year by Dealer, Make, and Model.

To create the filter values, do the following:

1. In each column except the last, assign a different id number to each unique entry in the column.

Dealer		Make		Model		Year
AutoMart	01	Ford	01	Taurus	01	1999
AutoMart	01	Ford	01	Taurus	01	1997
AutoMart	01	Ford	01	Windstar	02	1999
AutoMart	01	Toyota	02	Corolla	03	1999
AutoMart	01	Toyota	02	Corolla	03	1998
CarMania	02	Ford	01	Taurus	01	1998
CarMania	02	Ford	01	Mustang	04	1998
CarMania	02	Ford	01	Mustang	04	1996
CarMania	02	Honda	03	Civic	05	1999
CarMania	02	Honda	03	Accord	06	1998

2. Create new **integer** columns to hold the filter values; that is, insert one new column after every column in your worksheet, except after the first and last columns.

Dealer		Make			Model		Year
AutoMart	01	Ford	01		Taurus	01	1999
AutoMart	01	Ford	01		Taurus	01	1997
AutoMart	01	Ford	01		Windstar	02	1999
AutoMart	01	Toyota	02		Corolla	03	1999
AutoMart	01	Toyota	02		Corolla	03	1998
CarMania	02	Ford	01		Taurus	01	1998
CarMania	02	Ford	01		Mustang	04	1998
CarMania	02	Ford	01		Mustang	04	1996
CarMania	02	Honda	03		Civic	05	1999
CarMania	02	Honda	03		Accord	06	1998

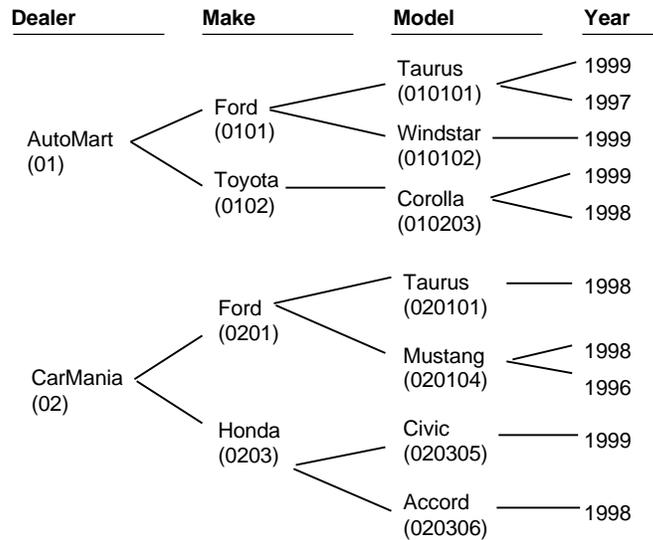
3. Name the new columns. We've chosen the names DealerMake and DealerMakeModel to convey the concept that the filter values identify the combination of the preceding selections, but you can use any name of your choosing.

Dealer		Make		DealerMake	Model		DealerMakeModel	Year
AutoMart	01	Ford	01		Taurus	01		1999
AutoMart	01	Ford	01		Taurus	01		1997
AutoMart	01	Ford	01		Windstar	02		1999
AutoMart	01	Toyota	02		Corolla	03		1999
AutoMart	01	Toyota	02		Corolla	03		1998
CarMania	02	Ford	01		Taurus	01		1998
CarMania	02	Ford	01		Mustang	04		1998
CarMania	02	Ford	01		Mustang	04		1996
CarMania	02	Honda	03		Civic	05		1999
CarMania	02	Honda	03		Accord	06		1998

4. Now, in each empty cell, combine the id numbers in the cells to the left to create a unique code representing the combination. For example, the DealerMake code in the first row would be 0101; the DealerMakeModel code, 010101.

Dealer	Make	DealerMake	Model	DealerMakeModel	Year
AutoMart 01	Ford 01	0101	Taurus 01	010101	1999
AutoMart 01	Ford 01	0101	Taurus 01	010101	1997
AutoMart 01	Ford 01	0101	Windstar 02	010102	1999
AutoMart 01	Toyota 02	0102	Corolla 03	010203	1999
AutoMart 01	Toyota 02	0102	Corolla 03	010203	1998
CarMania 02	Ford 01	0201	Taurus 01	020101	1998
CarMania 02	Ford 01	0201	Mustang 04	020104	1998
CarMania 02	Ford 01	0201	Mustang 04	020104	1996
CarMania 02	Honda 03	0203	Civic 05	020305	1999
CarMania 02	Honda 03	0203	Accord 06	020306	1998

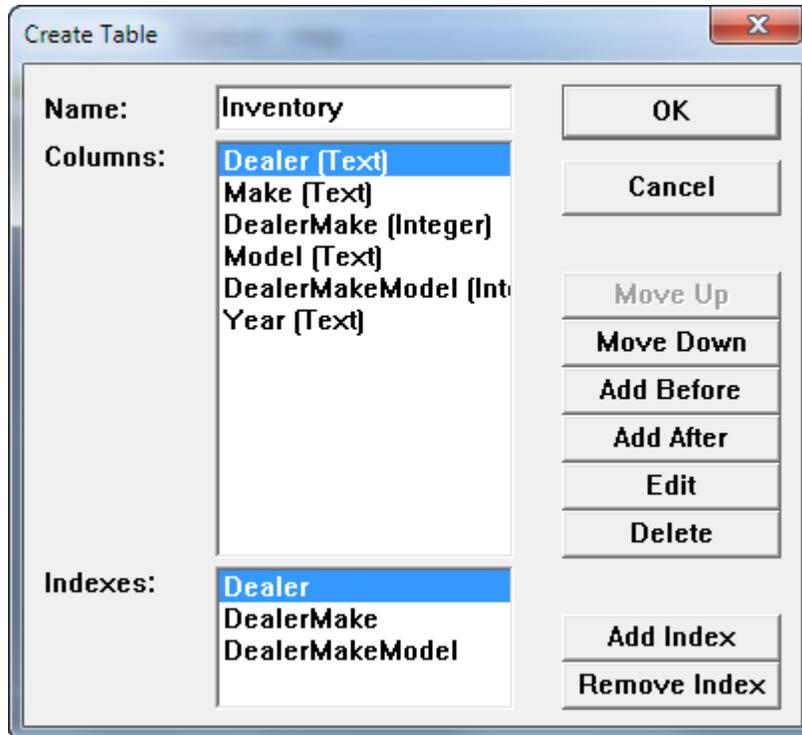
The coding scheme you choose to create the filter values is unimportant, as long as it uniquely identifies the combinations (decision paths) and can work as an unambiguous filter, as illustrated below.



Now that you have designed your scheme, you need to put it in place.

### Build the Support Table on the Desktop PC

Build the support table, "Inventory," as illustrated below. Remember, the columns for Dealer, DealerMake, and DealerMakeModel store filter values, so you must index them. Remember also that the columns DealerMake and DealerMakeModel store numeric codes, so they must be **integer** columns.



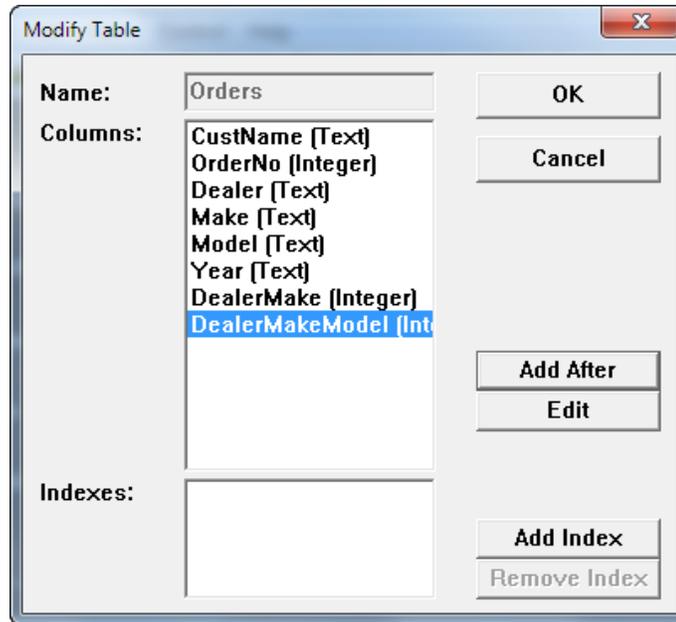
Create a form for the Inventory support table. Enter the records from your worksheet. Note: Don't enter the id numbers that you sketched in the Dealer, Make, or Model columns. Do enter the codes you created in the filter columns (DealerMake and DealerMakeModel), as illustrated below.

*Inventory Table*

Dealer	Make	DealerMake	Model	DealerMakeModel	Year
AutoMart	Ford	0101	Taurus	010101	1999
AutoMart	Ford	0101	Taurus	010101	1997
AutoMart	Ford	0101	Windstar	010102	1999
AutoMart	Toyota	0102	Corolla	010203	1999
AutoMart	Toyota	0102	Corolla	010203	1998
CarMania	Ford	0201	Taurus	020101	1998
CarMania	Ford	0201	Mustang	020104	1998
CarMania	Ford	0201	Mustang	020104	1996
CarMania	Honda	0203	Civic	020305	1999
CarMania	Honda	0203	Accord	020306	1998

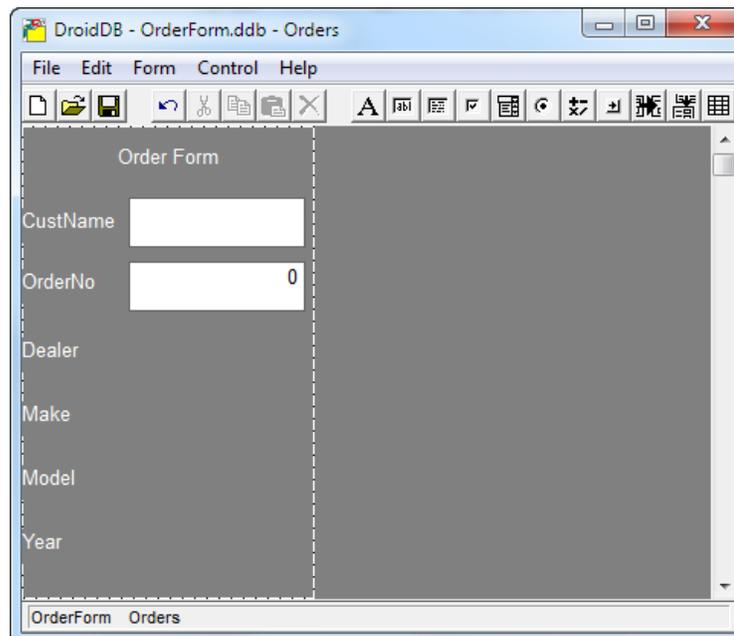
### Modify the Orders Table

In order for dependent drop-downs to work, there must be a matching column in the Orders table to store the filter values. It must have the same column name and data type as the corresponding column in the support table. That means you must add DealerMake and DealerMakeModel to the design of your Orders table, as illustrated below.



### Specify the Drop-Down Controls in the Application Form

Remember, our goal is to create intelligent drop-downs. If your original form design for the application used simple edit controls for these fields, delete them from the form.



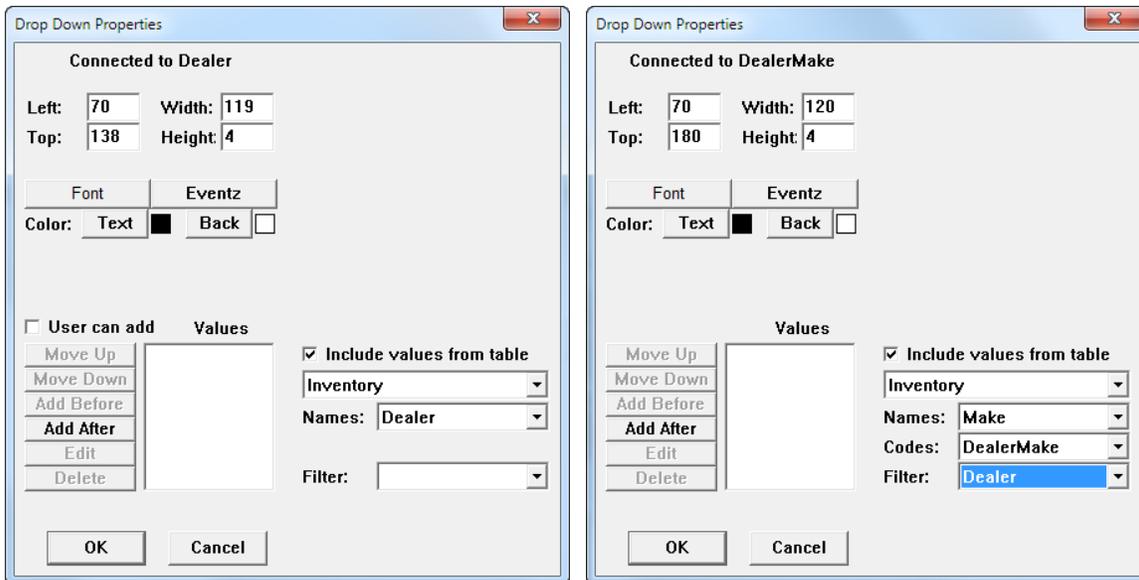
Before recreating the controls as drop-downs, you may find it helpful to plan your strategy on paper first, using a chart similar to the one shown below. Keep in mind that this is not a table to enter into the

computer – it's just a chart you can refer to when specifying the drop-down control properties for the Orders form. Each line in this chart represents one drop-down control you will create in the form.

Control Label	Connected to	Include values from table	Names	Codes	Filter
Dealer	Dealer	Inventory	Dealer		
Make	DealerMake	Inventory	Make	DealerMake	Dealer
Model	DealerMakeModel	Inventory	Model	DealerMakeModel	DealerMake
Year	Year	Inventory	Year		DealerMakeModel

The controls are listed in the order you create them (Dealer, Make, Model, Year). Note that a control's Filter column is the same as the previously created control's "Connected to" column.

Create the drop-down controls as you would normally, using the specifications listed in the chart for the control properties. The two screen shots below illustrate how the specifications in the first two lines in the chart are applied to the properties for the first two controls.



Test your new application on the Android device to make sure that the drop-downs display the correct options. The illustration below shows how the finished order form might look on the Android device.

Order Form

CustName

OrderNo

Dealer

Make

Model

Year

## Creating Navigational Drop-Down Lists

The previous topics explained how to create drop-down lists that enable users to select a value to store in a new or modified record. This topic explains how to create drop-down lists that enable users to quickly find existing records by picking a desired value.

Navigational drop-down lists are dynamically composed of values stored in a column in the application table. When the user selects a value from the list, the form displays the first record in the table that has the selected value.

Navigational drop-downs can be associated with any text, memo, or numeric column that has been indexed. They cannot be associated with date/time, yes/no, or picture columns; nor with global variables.

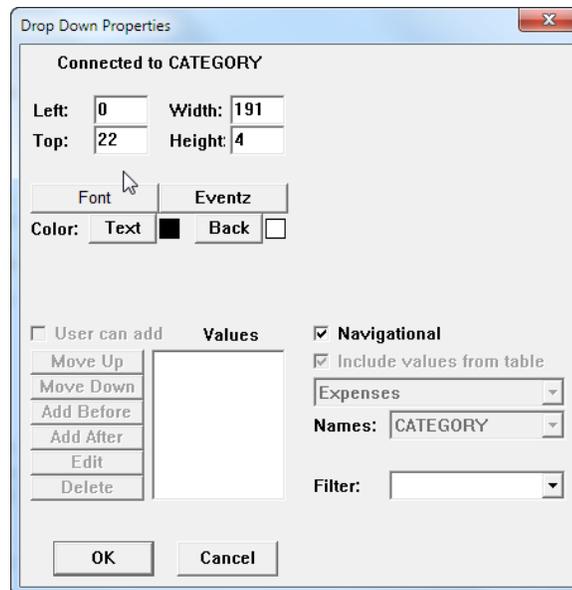
To create a navigational drop-down list:

1. Select **Control > Drop Down**, or click the **Drop Down** button , or right-click where you want the control to appear and select **Drop Down**.

DroidDB displays the Select Columns dialog box.

2. Select the table column that contains the values you want users to search by. Remember, this must be an indexed text, memo, or numeric column. Click **OK**.
3. In the Drop-Down Properties dialog, select the **Navigational** checkbox.

DroidDB then displays additional fields in the lower right corner of the Drop Down Properties dialog.



4. Complete the additional fields:
  - In the first two new fields (directly under the "Include values from table" checkbox), leave the default values in place. The first field displays the name of the table that the form is built on (in the example above, "Expenses"). The second field, "Names," displays the name of the table column you selected in Step 2.

- **Filter:** Complete this field if you wish to filter the options displayed in the drop-down list by the user's selection for another form control. Refer to the topic, "Creating Dependent Drop-Down Lists," on page 139 for details.
5. Complete the options in the upper left corner of the dialog box to position and style the appearance of the control as desired.

**Note**

The options in the lower-left corner of the Drop Down Properties dialog box do not apply to navigational drop-down lists. If you select one of these options, DroidDB hides the options that do apply to navigational drop-down lists.

6. Click **OK** on the Drop Down Properties dialog box.
7. Drag the drop-down list to the correct location, and resize it if necessary.

## Creating Calculated Fields

You can incorporate arithmetic or text operations—and even advanced functions—right into your form. When a user creates or displays a record, DroidDB automatically applies the embedded formula (called an “expression”) to the current record's values and displays the results in the calculated field. You specify whether DroidDB performs the calculations on command, or automatically whenever the record is displayed. You also specify whether or not DroidDB saves the results in the table or a global variable. The results can be numeric, text, or date/time.

For example, the sample Expense Tracker application that you worked with in the Quick Tours included a calculated field control that automatically calculated a 15% tip on the amount of the expense. It used the following expression:

$$\text{Amount} * 15 / 100$$

Here are a few other examples of calculated field expressions:

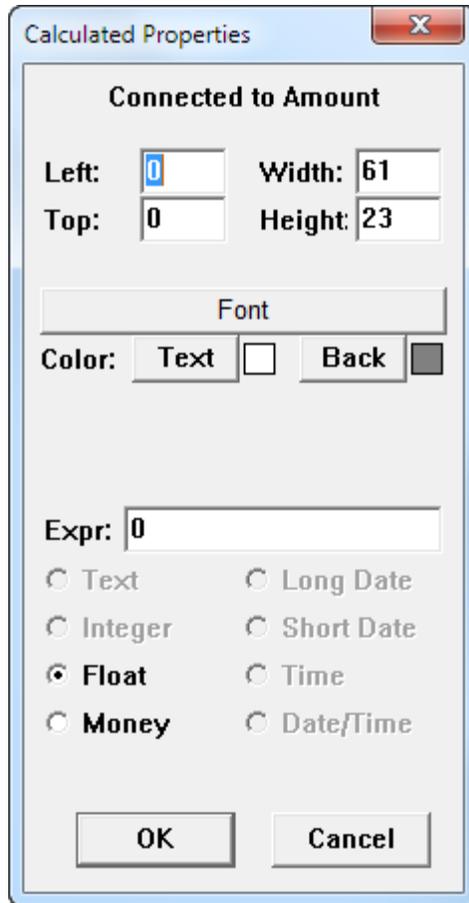
This Expression	Returns
<code>UnitPrice * Quantity</code>	3.00 when the value in the UnitPrice column is 1.50 and the value in the Quantity column is 2
<code>"First Name" &amp; ' ' &amp; "Last Name"</code>	'Jane Smith' when the value in the First Name column is 'Jane' and the Last Name column is 'Smith'
<code>@now + (5 * @days)</code>	The date five days from today

This topic explains how to add a calculated field control to your form. You'll find detailed instructions for defining expressions in “Part III: Expressions and Functions,” later in this guide.

To create a calculated field:

1. Select **Control > Calculated**, or click the **Calculated** button , or right-click where you want the control to appear and select **Calculated**.
2. DroidDB asks if you want to store the results of the calculation.
  - If you click **Yes**, DroidDB asks you to select the table column or global variable where the results will be stored.
  - If you click **No**, DroidDB will display the results of the calculation in the application's form, but will not store them.

DroidDB displays the Calculated Properties dialog box.



The Calculated Properties dialog box contains the following fields:

Field	Description
Connected to [column or variable]	The table column or global variable to which the control relates – cannot be changed. (If you did not specify a column or variable to store the results, the Label field appears in the Calculated Properties dialog box instead.)
Label	Optional field for recording comments in the Calculated Properties dialog box. Your entry here does not appear on the application form.
Left	The initial horizontal position of the left side of the control.
Top	The initial vertical position of the top of the control.
Width	The initial width of the control.
Height	The initial height of the control.
Font	The font, size and style of the type in the calculated field. Click to modify. Refer to the topic, "Specifying Font Properties for a Control," on page 174 for instructions.
Text	The color of the type in the calculated field.

Back	The color of the background in the calculated field.
Expr	The expression, or formula, that DroidDB will apply to return a result. For more information, refer to “Part III: Expressions and Functions,” later in this guide. Tip: Double-click the Expr field to display a larger text area in which to enter the expression.
[Result options]	Specify the display format for the results. (If you selected a column or variable in which to store the results, only formats suitable for that datatype are available.) If you check Money, the resulting value is displayed with two decimal places, e.g., 15.28 Note: The display format associated with each date/time option is determined by the Date/Time Settings on the device that runs the form. Refer to the topic, “Edit Box Date/Time Formats,” on page 128 for details.

3. Enter the expression in the Expr field of the dialog box, make any other changes required in the Calculated Properties dialog box, then click the **OK** button.
4. Drag the calculated field to the correct location on the form, and resize it if necessary.
5. If necessary, create a label to identify the calculated field.
6. If you want DroidDB to perform the calculation automatically, turn on Auto Recalc by selecting **Form > Auto Recalc** so that a checkmark appears next to the option. (Refer to the topic, “Enabling the Auto Recalc Feature, on page 187 for details.)

## Creating Command Buttons

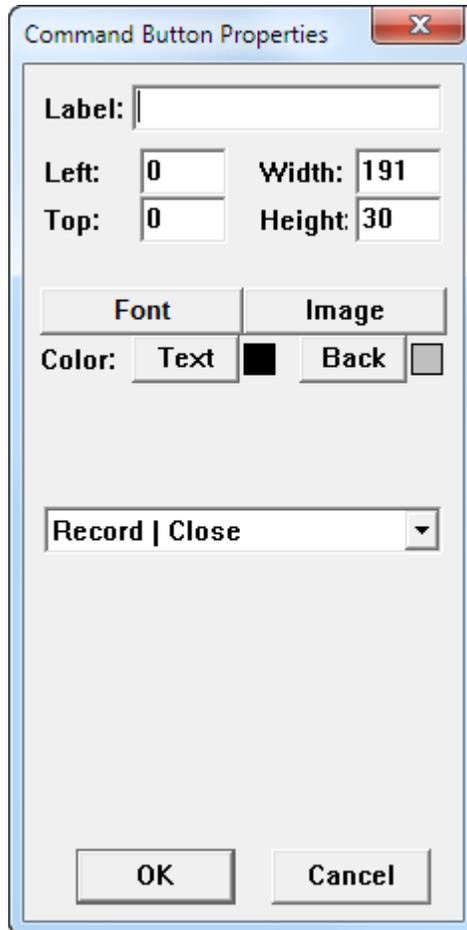
A command button is a button on a form that users click to execute a task. For example, you can create a command button that serves as a shortcut for menu picks, duplicates the current record, launches a macro, or opens another form in the application.

Each of the commands in the DroidDB application menus (e.g., Record > Insert), as well as a library of special commands can be assigned to a command button. A list of commands that can be assigned to a command button is provided at the end of this topic.

To create a command button:

1. Select **Control > Command Button**, or click the **Command** button , or right-click where you want the control to appear and select **Command**.

DroidDB displays the Command Button Properties dialog box.



Complete the properties as described below.

Field	Description
Label	The label to appear on the button.
Left	The initial horizontal position of the left side of the button.

Top	The initial vertical position of the top of the button.
Width	The initial width of the button.
Height	The initial height of the button.
Font	The font, size and style of the type used as the label on the button. Click to modify. Refer to the topic, "Specifying Font Properties for a Control," on page 174 for instructions.
Image	Optional. The filename of an image (.bmp or .jpg) to be displayed on the face of the button. The image must be stored in the application folder on the Android device. <b>Tip:</b> You can gang together a number of buttons on a form to create an apparently single image with hotspots.
Text	The color of the type that appears on the button.
Back	The color of the button.

- From the command drop-down list, select the action that DroidDB will perform when the user clicks your command button. Brief descriptions of the commands are provided in the table, "DroidDB Commands Frequently Attached to Command Buttons," which follows.

Depending upon your selection, DroidDB may display additional options at the bottom of the Command Button Properties dialog box. The following table directs you to instructions for completing those options.

**Note**

The following chart describes the DroidDB commands that you would most likely attach to a command button. (Additional commands appear in development environment drop-down list, but they are intended primarily as steps in a macro sequence and their use on command buttons is discouraged. You can find additional information about all commands in the Command Reference beginning on page 230.)

- Once you have completed the options in the Command Button Properties dialog, click **OK**.
- Drag the command button to the correct position and resize it if necessary.

**DroidDB Commands Frequently Attached to Command Buttons**

<u>Command</u>	<u>Does this</u>	<u>For Info about options...</u>
Record   First	Displays the first record	
Record   Previous	Displays the previous record	
Record   Next	Displays the next record	
Record   Last	Displays the last record	
Record   Search	Moves the display to the first record whose value matches a search value. Control can prompt user for search value, or you can have the form get the search value from a global variable. Search is performed on whichever column the table is currently sorted on. Search can be either for exact match or closest (if no match, next greatest).	See page 250

Record   Save	Saves the current record	
Record   Insert	Creates a new record	
Record   Duplicate	Duplicates the current record (Note: this command is not available in the application menu bar)	
Record   Delete	Deletes the current record	
Record   Close	Closes the application	
Option   Filter   Off	Displays all records in table	
Option   Filter   Predefined	Runs a predefined record filter	See pages 182 and 246
Option   Import	Imports contents of a comma delimited ASCII text file into the current table. You can choose whether or not to include table column names.	See page 247
Option   Export	Exports records from the current table or filtered list to a comma delimited ASCII text file. You can choose whether or not to include table column names.	See page 245
Option   Clear	Deletes all records in the table	
Option   Recalc	Recalculates calculated field values in the current record	
Option   About	Displays information in the About box	
Communicate   GPS	Gets current location	See page 237
Barcode	Activates barcode scanner and stores scanned input	See page 236
Dial	Dials a phone number stored in the current record or global variable	See page 238
mEnable Synchronize	Synchronizes all eligible local tables with tables in a remote database server	See page 241
Report	Displays records in a DroidDB report layout	See page 251
Run external	Launches another executable program and (optionally) passes a value from the current record to the second program	See page 253
Run form	Launches another DroidDB form in the same application.	See page 254
Run macro	Runs a macro. You must have already created and named the macro as part of the current DroidDB form.	See page 255
Sort by	Sorts records by values in an indexed column	See page 263

SMS Receive Message	Launches a DroidDB form upon receipt of an SMS instant text message that begins with the string .ddb. and optionally saves the incoming message to a variable	See page 262
SMS Send Message	Sends an SMS instant message from the application to a specified phone number	See page 262
Timer	Starts a specified macro after a delay of some number of seconds	See page 265

**Note**

The following commands appear in the development environment drop-down list for command buttons. However, they are intended primarily as steps in a macro sequence, and their use as command buttons is discouraged.

- Stop macro
- Return from macro
- Message box
- Assign
- Skip
- Set focus
- Show/Hide control
- Select tab
- Play sound
- Sleep

## Creating Scribble Boxes

A scribble box lets users draw, save, and view freehand images such as simple sketches and signatures.

A scribble box can be read-only or read/write. It must be associated with a column defined for the Picture data type.

To create a scribble box:

1. Select **Control > Scribble** or click the **Scribble** button , or right-click where you want the control to appear and select **Scribble**.

DroidDB displays the Select Column dialog box.

2. Click the column in which the scribbles will be stored, then click **OK**.

DroidDB displays the Scribble Properties dialog box, which contains the following fields:

Field	Description
Connected to [column]	The table column to which the control relates – cannot be changed.
Left	The initial horizontal position of the left side of the control.
Top	The initial vertical position of the top of the control.
Width	The initial width of the control.
Height	The initial height of the control.
Events	Optional. Select macros to be triggered when the user puts the focus on and/or moves the focus away from the control. Refer to “Triggering Macros with Events,” on page 227.
Read Only	If checked, users cannot draw in the scribble box.

3. Make any necessary changes to the scribble box properties, then click **OK**.
4. Drag the box to the correct position, then resize it if necessary.
5. Create a label to identify the scribble box, if desired.

## Using an Image Control to Place a Graphic in the Form Design

An image control is a box that displays pictures (.bmp, .gif, or .jpg files). This topic explains how to use an image control to place a static design element on the form, such as a company logo.

1. Put the graphic file (a .bmp, .gif, or .jpg) in the application folder on the Android device. (The application folder has the same name as your application and resides at the top-level of the Android device directory structure).

2. Select **Control > Image**, or click the **Image** button , or right-click where you want the graphic to appear and select **Image**.

3. DroidDB displays a message asking whether or not the image depends on a value in a table column or global variable. Select **No**.

DroidDB displays the Image Properties dialog box.

4. In the **Image** field, select the graphic you want to appear on the form.
5. Make any changes necessary to the remaining properties, then click **OK**.

Field	Description
Left	The initial horizontal position of the left side of the control.
Top	The initial vertical position of the top of the control.
Width	The initial width of the control.
Height	The initial height of the control.
Events	Optional. Select macros to be triggered when the user puts the focus on and/or moves the focus away from the control. Refer to “Triggering Macros with Events,” on page 227.

6. The graphic appears in the development window as it will appear on the form. Drag the image box to the correct position and resize it if necessary.

## Creating an Image Control to Take Pictures with the Camera

If a camera is integrated into the Android device, an image control can be used to capture photos.

The user taps the image control and takes the photo using the device’s camera application. If you have associated the control with a picture column, DroidDB stores the photo in the record. When the record is synchronized to the desktop, it will be an OLE Object (bitmap) in the desktop database. If you have instead associated the control with a text column or global variable, DroidDB puts the photo file in the application folder on the Android device and stores the filename in the record. DroidDB automatically assigns the filename using the convention: *deviceName\_dateTimeTaken*.

To create an image control to take and store photos:

1. Select **Control > Image**, or click the **Image** button , or right-click where you want the control to appear and select **Image**.
2. DroidDB displays a message asking whether the image depends on a value in a table column or global variable. Select **Yes**.
3. DroidDB displays the Select Column dialog box, listing available text and picture columns, and global variables. Select a column or variable to store the image information. The type of column you choose depends upon where you wish the image files to be stored, either:
  - Select a picture column to store the images in the table, or
  - Select a text column or a global variable to store the image filenames. The image files themselves will be automatically stored in the application folder on the Android device.

When you have made your selection, Click **OK**. DroidDB displays the Image Properties dialog box.

Field	Description
Connected to [column or variable]	The column or variable that stores the image information – cannot be changed.
Left	The initial horizontal position of the left side of the control.
Top	The initial vertical position of the top of the control.
Width	The initial width of the control.
Height	The initial height of the control.
Events	Optional. Select macros to be triggered when the user puts the focus on and/or moves the focus away from the control. Refer to “Triggering Macros with Events,” on page 227.
Image Selection on Click	This option enables/disables the camera. It must be selected in order to use a camera with the control.

4. Make any changes necessary to the Image control properties and click **OK**.
 

**Note:** Be sure that the “Image Selection on Click” option is checked.
5. Drag the box to the correct position, then resize it if necessary.
6. Create a label to identify the image control, if desired.

## Creating a Jump Button

Jump buttons link forms. They display or create records in related tables. For example, you could have a jump button on a Customer form that opens an Orders form displaying the customer's orders.

Jumps utilize a search key value to associate the current record (the one displayed in the first form) with the record to jump to in the second form. Typically, the "jump from" and "jump to" forms are built on separate tables that both have a column with the same name and datatype: this is the "key column." Matching values in the key columns indicate related records. Alternatively, the key value in the "jump from" form may be from a global variable.

When the user clicks a jump button, DroidDB reads the current record's value in the key column or global variable. It opens the "jump to" form and displays the first record in the "jump to" table that has a matching value in the key column. If a matching key value does not exist in the "jump to" table, you can specify that DroidDB creates a new record with the key value filled in, or display a message that no related records exist. Alternatively, you can specify that DroidDB always creates a new record in the "jump to" table with the key value filled in, regardless of whether or not matching records exist.

Say you have two forms, ORDERS and CUSTOMERS, with tables of the same name. Both ORDERS and CUSTOMERS have a table column for customer numbers (CUSTNUM). CUSTNUM is the key column. You could include a jump button on the CUSTOMERS form that would enable users to jump to the ORDERS form. Referring to the tables below, if custnum 2000 were currently displayed on the form, order A1652 would appear when the user clicked the jump button. On the other hand, if a user were on Custnum 3000 and clicked the jump button, there would be no matching records to jump to...so DroidDB would create a new order record with Custnum 3000 already supplied.

CUSTOMERS			
CUSTNUM	LASTNAME	FIRSTNAME	PHONE
1000	Smith	John	(555) 555-0000
2000	Jones	Mary	(555) 555-9988
3000	Green	Bob	(555) 555-7540

ORDERS	
CUSTNUM	ORDERNUM
1000	A1701
1000	A1351
2000	A1652

### Tips

You can incorporate Jumps in a macro program. Refer to the section, "Macros and Events" beginning on page 212 for details.

### Prerequisites

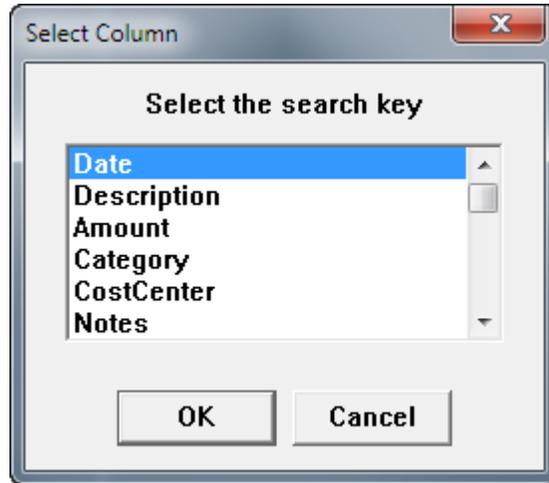
The forms and tables must belong to the same application.

The key column in the destination table must be indexed. (In the example above, you would have to index the CUSTNUM column in the ORDERS table before adding the jump button to the Customers form.) See "Indexing Columns," on page 181 for instructions.

To create a jump button:

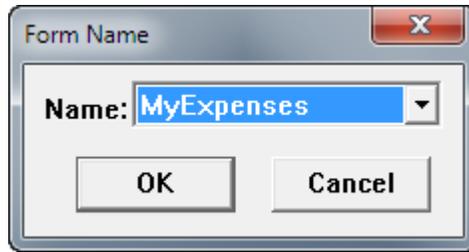
1. Select **Control > Jump**, or click the **Jump** button , or right-click where you want the control to appear and select **Jump**.

DroidDB displays the Select Column dialog box.



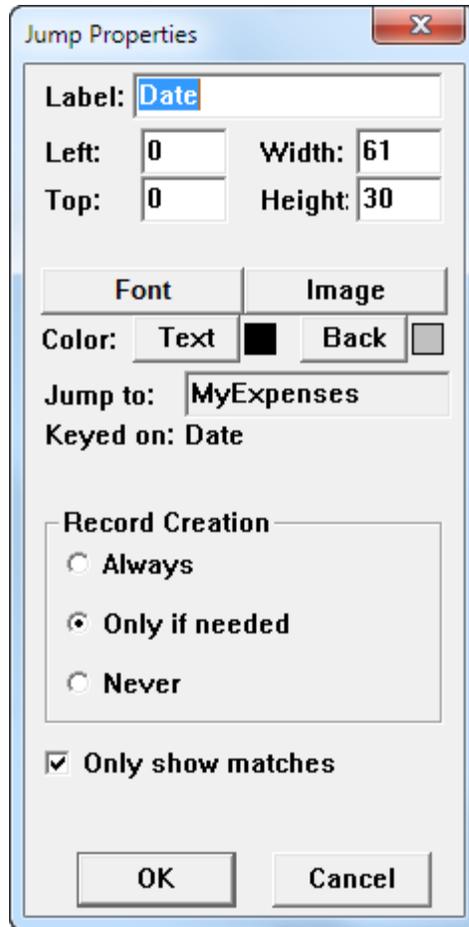
2. Select the column or global variable in the current form (the "jump from" form) that will contain the search key, then click **OK**. (The search key is the value that DroidDB will use to match records in the "jump to" table.)

DroidDB displays the Form Name dialog box.



3. Select the form that will be the destination for the jump button, and click **OK**.
4. If you selected a global variable in step 2, the Select Column dialog box appears. Select the key column for the "jump to" form, and click **OK**.

DroidDB displays the Jump Button Properties dialog box.



The Jump Properties dialog box contains the following fields:

Field	Description
Label	The text that appears on the button in the form. Column name is the default.
Left	The initial horizontal position of the left side of the control.
Top	The initial vertical position of the top of the control.
Width	The initial width of the control.
Height	The initial height of the control.
Font	The font, size and style of the type for the label on the button. Click to modify. Refer to the topic, "Specifying Font Properties for a Control," on page 174 for instructions.
Image	Optional. The filename of an image (a .bmp or .jpg) to be displayed on the face of the button. The image must be stored in the application folder on the Android device. Tip: You can gang together a number of buttons on a form to create an apparently single image with hotspots.
Text	The color of the type that appears on the button.
Back	The color of the button.
Jump To	The form that DroidDB opens when the user clicks the jump button (cannot be changed).

Keyed on [variable/column]	The matching columns, or variable/column combination, that the jump is keyed on (cannot be changed).
Record Creation	Action that DroidDB takes when opening second form:  <b>Always</b> - Always create a new record. If you select the "Only show matches" checkbox, DroidDB automatically fills in the value from the key column or variable.  <b>Only if Needed</b> - Create a new record only if there are no existing records with a matching value in the key column. If you select the "Only show matches" checkbox, DroidDB automatically fills in the value from the key column or variable.  <b>Never</b> - Do not create a new record. Display first record in "jumped to" table that has a matching value in the key column. If no matches are found, display messages, "Record not found."
Only show matches	If checked, DroidDB turns on a filter for records in the "jumped to" table, so that only records that have a matching value in the key column are retrieved for display. If unchecked, all records in the "jumped to" table are retrieved for display.

5. In the Label field, replace the column name with a label for the button (if desired). Make any other changes required in the jump properties dialog box, and click the **OK** button.
6. Drag the button to the desired location on the form.

## Creating a Lookup

A lookup control retrieves and displays a value from a related record in another table (the "lookup table"). For instance, you could design an order form in which DroidDB looks up the price when the user enters a part-number.

You can specify whether the retrieved value is saved with the record, written temporarily to a global variable, or merely displayed in the lookup control on screen.

### Prerequisites

DroidDB matches the lookup record to the record currently displayed in the form using a search key value that is common to both. Typically the table that the form is built on and the lookup table each have a column with the same name and datatype that stores the key values. Alternatively, a global variable may be used to store the key value for the current record in the form. For example, you might have a calculated field in the form that writes a temporary value for the current record to a global variable. DroidDB would compare that computed value to values stored in the lookup table's key column to identify the desired record.

In all cases, the key column in the lookup table must be indexed. (See the topic, "Indexing Columns," on page 181 for instructions.)

If you want to save the lookup value with the record, the form's table must have a column with the same name and datatype as the column that stores the lookup values in the lookup table.

The form and tables must be in the same application.

### Tip

There are one or two options for making it possible to add and maintain data in a lookup table:

- You can build a form on top of the lookup table, and add a "Run Form" command button on the primary form to open it. (The primary form is the one that opens automatically when the user launches the application. It has the same name as the application.)
- Business Edition only: You can maintain a version of the lookup table on the desktop PC and update its Android device counterpart via synchronization.

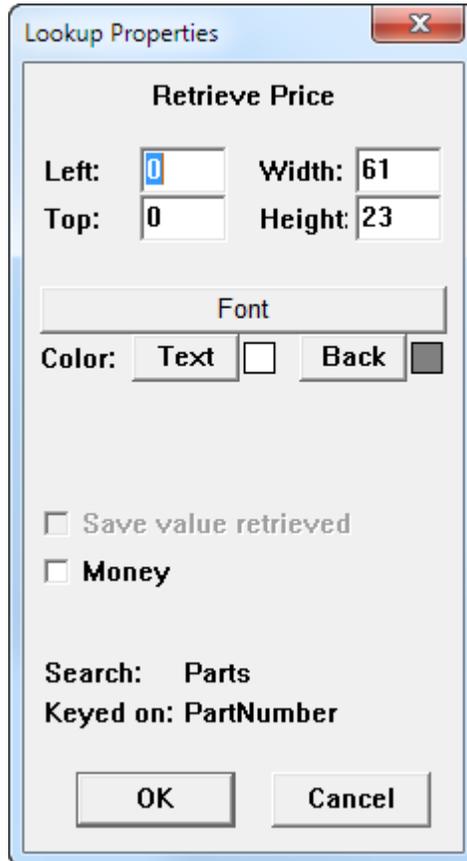
To create a lookup control:

1. Select **Control-Lookup**, or click the **Lookup** button , or right-click where you want the control to appear and select **Lookup**.
2. DroidDB displays the Select Table dialog box. Select the table that stores the lookup values, and click **OK**.
3. DroidDB displays the Select Column dialog box. Select the column or global variable that contains the form's search key value, then click **OK**.
4. If you selected a global variable in the previous step, DroidDB displays the Select Column box. Pick the column in the lookup table that stores the search key values, and click **OK**.
5. DroidDB asks if you want the lookup value saved:
  - Respond **No** if you want the lookup value displayed in the form but not saved.
  - Respond **Yes** if you want the lookup value saved when the user saves the record. DroidDB asks where to store the lookup value. Select a table column in the form's table or a global

variable. (If you select a table column, DroidDB assumes that you are retrieving values from a matching table column in the lookup table.)

- If, in the previous step, you indicated either that you do not want to save the lookup value or that you do want to save it in a global variable, DroidDB displays the Select Column dialog box. Select the column in the lookup table that contains the values you want retrieved from the lookup table.

DroidDB displays the Lookup Properties dialog box for the lookup control.



The Lookup Properties dialog box contains the following fields:

Field	Description
Retrieve [column/variable]	The table column or global variable that contains the lookup value to be displayed by the control – cannot be changed.
Left	The initial horizontal position of the left side of the control.
Top	The initial vertical position of the top of the control.
Width	The initial width of the control.
Height	The initial height of the control.
Font	The font, size and style of the type that appears in the lookup field. Click to modify. Refer to the topic, "Specifying Font Properties for a Control," on page 174 for instructions.

Text	The color of the type that appears in the lookup field.
Back	The color of the background in the lookup field.
Save value retrieved	If checked, value in lookup field will be saved with the record. If unchecked, value will be displayed but not saved. (Whether or not this option is checked depends upon your response in Step 5 above. You cannot change this property here.).
Money	If the lookup value you specified is stored as a floating point number, the Money checkbox displays. If Money is checked, entries are displayed with two decimal places.
[Date/time options]	If the column is associated with dates/times, four options are available for the type of data that can be displayed: Long Date, Short Date, Time, or Date/Time. Note: The display format associated with each option is determined by settings on the device running the form.  Refer to the topic, "Edit Box Date/Time Formats," on page 128 for details.
Search: [tablename]	The name of the lookup table – cannot be changed.
Keyed on: [columnname/variable]	Name(s) of the keys that associate the records in the two tables – cannot be changed

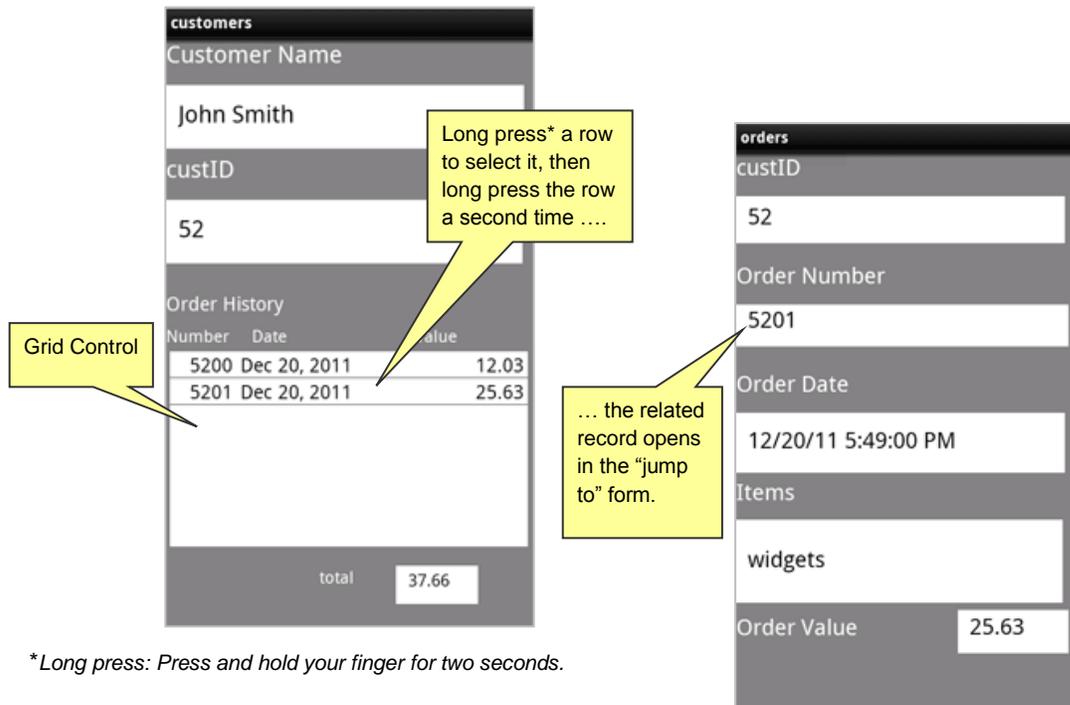
7. Make any changes required to the lookup properties, then click **OK**.
8. Drag the lookup control to the desired location on the form.
9. If necessary, create a label to identify the lookup field.

## Creating a Grid Control

A grid control is a scrollable window within a form that displays a list of all of the records in the form's table, or related records from another table. You can choose which record values are displayed, and how the records are sorted. You can even display summary values for the total number of records in the grid and the sum of the values in a grid column.

### How the Grid Works When Displaying Related Records from a Different Table

When used to display related records from a different table, grid controls combine and expand the functions of lookup and jump controls. (See illustration below.) While lookups display just one value from the first related record in another table, grids can display multiple values for all related records in the other table. If the user activates a row in the grid window, DroidDB displays the selected record in another DroidDB form (just like a jump button). You can also turn off the jump, so that the grid is simply a display.



### How the Grid Works When Displaying Records from the Current Table

You can also use a Grid Control to display all records in the current form's table, providing a convenient record list view. If the user does a double long press on a row in the grid, DroidDB displays the selected record in the current form. If you want the grid to display a simple record listing, you have the option of turning off this jump behavior.

### Prerequisites if Displaying Related Records from a Different Table

The main table must have one column that matches a column in the other table (which we will call the "jump-to" table). Both columns must have the same column name and store the same data type. This column is referred to as the "key" column, since DroidDB uses it to match records in the two tables. (In the example above, CustID is the key column.)

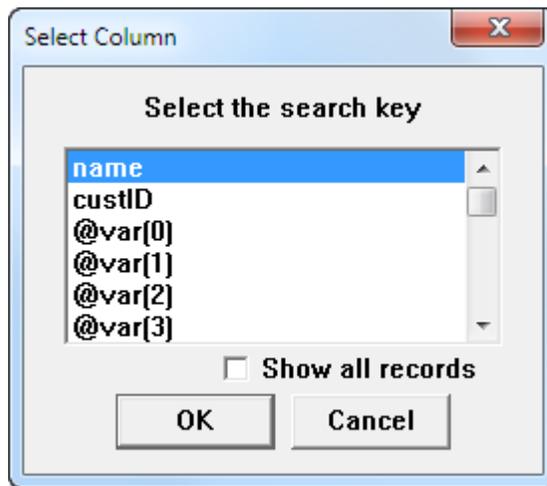
Alternatively, a global variable may be used to store the key value. For example, you might have a calculated field in the main form that writes a temporary value for the current record to a global variable. DroidDB would compare the computed value to values stored in the jump-to table's key column to identify related records.

In both cases, the key column in the jump-to table must be indexed. (See the topic, "Indexing Columns," on page 181 for instructions.)

To create a grid control:

1. Select **Control-Grid**, or click the **Grid** button , or right-click where you want the control to appear and select **Grid**.

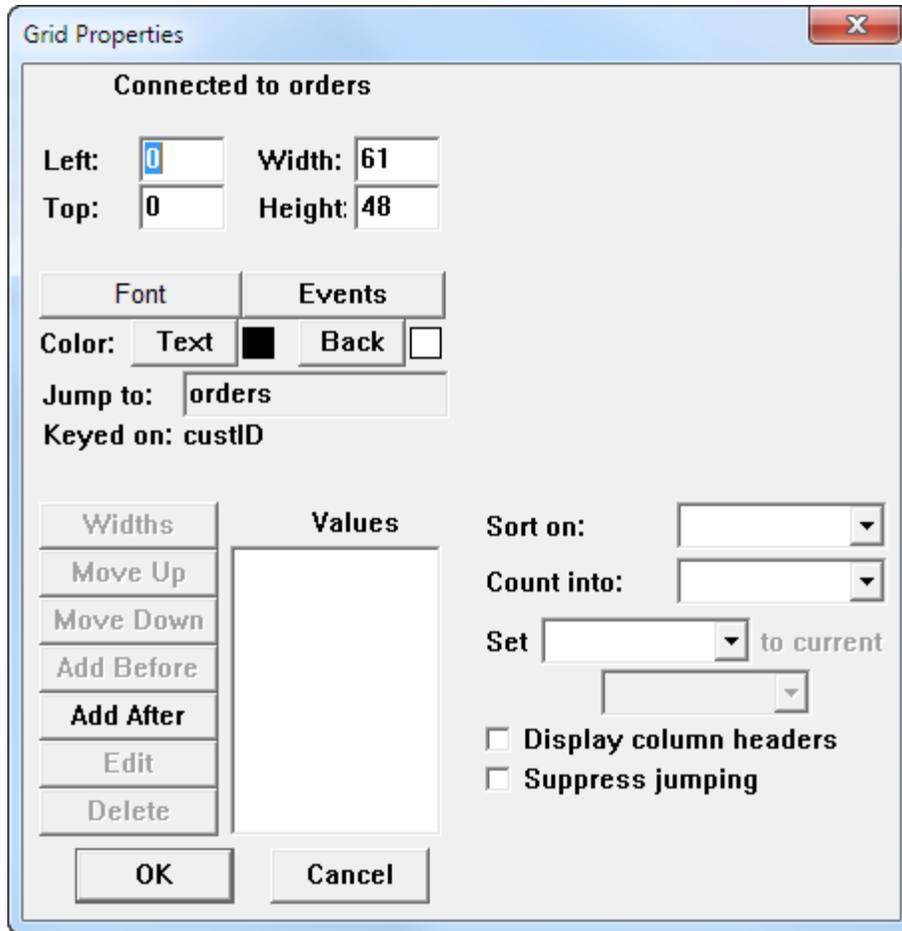
DroidDB displays the Select Column dialog box.



Your selection depends upon whether you want the grid to display all records from the current form's table, or related records from another table:

- To display all records from the form's table, select **Show all records**, and click **OK**.
  - To display records from a related table, select the form's "key" column or a global variable that will hold the value used to identify matching records in the related table. Click **OK**.
2. If you selected a search key in the previous step, DroidDB displays the "Form Name" box. Select the DroidDB form that will be opened when the user activates a row in the grid control (i.e., the "jump to" form). Click **OK**.
  3. If you selected a global variable to store the "jump-from" form's key value, DroidDB asks you to select the key column in the jump-to table. Select it and click **OK**.

DroidDB displays the Grid Properties dialog box.



Edit the Grid properties as necessary:

Field	Description
Connected to [table]	The application or jump-to table – cannot be changed.
Left	The initial horizontal position of the left side of the control.
Top	The initial vertical position of the top of the control.
Width	The initial width of the control.
Height	The initial height of the control.
Font	The font, size and style of the type displayed inside the grid window. Click to modify. Refer to the topic, "Specifying Font Properties for a Control," on page 174 for instructions.
Events	Optional. Select macro to be triggered when the row selected in the control is changed. Refer to "Triggering Macros with Events," on page 227.
Text	The color of the type that appears in the grid window.

Back	The color of the background in the grid window.
Jump To	<i>Shown only if you selected a search key in Step 1.</i> The form that DroidDB opens when the user does two long-presses on a row in the grid window (cannot be changed).
Keyed on [column or variable]	<i>Shown only if you selected a search key in Step 1.</i> Table column or global variable that jump is keyed on (cannot be changed).
Values	Columns to appear in the grid window. See Step 4 below.
Sort on	Optional. Column containing the values DroidDB will use to sort the rows in the grid (column need not be displayed in the grid window).
Count into	Optional. Global variable to store count of records retrieved and displayed in the grid.
Set [global variable] to current [table column]	Optional. Useful if row selection fires an event. When the user long-presses a row in the grid, the value stored in the specified table column for the row will be assigned to the global variable. That way, the processing of the event will know which row was just selected.
Display column headers	If checked, the table column names appear as column headers in the grid window.
Suppress jumping	If checked, the grid is display-only. If unchecked, users can select a row in the grid to open that record.

4. For each table column you want to appear in the grid window:
  - Click the **Add After** button (or **Add Before** if available and you prefer). DroidDB displays the Grid Column dialog box.



- In the **Name** field, select a column whose values you want to appear in the grid window.
  - Depending upon the type of data stored in the column, DroidDB displays options for displaying the data in the grid, such as whether or not you wish display numbers as money values, or the format for date/time values. For numeric columns, the **Sum into** field is available -- you can select a global variable to store the sum of the values in the grid column. (For example, see the "Total" field shown in the illustration at the beginning of this topic.) Select the desired options (if any), and click **OK**.
5. DroidDB will display the columns in the grid from left to right in the same order they are listed in the Values box from top to bottom. You can change the order by selecting a name in the Values box and clicking **Move Up** or **Move Down**. You can also remove a column from the grid by selecting its

name and clicking **Delete**. And, you can change the settings you specified in the Grid Column box by selecting the name and clicking **Edit**.

6. When you have finished specifying the Grid properties, click **OK** to close the Grid Properties dialog box.

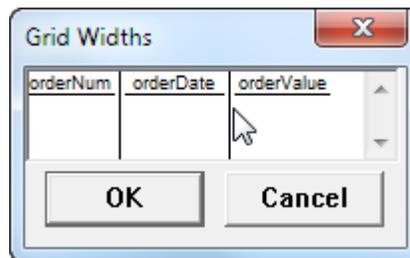
DroidDB displays the grid, allotting equal width to each column.

**Tip**

DroidDB allows you to modify the widths of the columns within the grid window (described in the next step). You may find it easiest if you first make the window itself the desired size, then modify the column widths.

7. If you wish to modify the widths of the columns in the grid window:

- Double-click the grid control itself.  
DroidDB redisplay the Grid Properties dialog box.
- Click the **Widths** button in the Grid Properties dialog box.  
DroidDB displays the Grid Widths dialog box.



- Point the cursor at a vertical line dividing two columns and drag to the left or right.
  - When you have resized the columns as desired, click **OK**. Then click **OK** to close the Grid Properties dialog box.
8. Drag the window to the desired location.
  9. If you assigned summary values to global variables, create a calculated field control to display each one.

# Modifying Control Properties

## Changing Control Properties

Once you have created a control, you can change some of its properties, but not all.

The only properties you **cannot** change are:

- The type of control. For example, you cannot change radio buttons into a drop-down list.
- The table column or global variable to which the control is associated.

Most of the other properties of a control can be changed.

### Note

If you do need to change a control's type, column, or other properties that cannot be changed, delete the existing control and create a new one.

To change a control's properties:

1. Double-click the control (or click the control once and select **Edit > Property**, or right-click the control once).

DroidDB displays the control's Property dialog box.

2. Make any changes required, then click **OK** on the Property dialog box.

DroidDB changes the control's properties.

### Tip

You can change properties for several controls at once (refer to the topic, "Changing Properties for Multiple Controls," on page 175).

## Specifying Font Properties for a Control

The Properties dialog boxes for most types of controls include a Font button that enables you to assign a font, size, and style (bold, italic, and underline) to the characters in the control. Using this feature, you can give your form a more personalized look, as well as add clarity by emphasizing important elements.

### Note

If you do not assign specific font properties to a control using the Font properties dialog box, DroidDB automatically uses the default settings, which are Arial, 10 point, with no styles.

To assign font properties to a control:

1. In the Properties dialog box, click the Font button



DroidDB displays the Font dialog box.



2. Assign font properties to the characters displayed by the control:
  - **Font:** Choose between Courier New (a monospaced font), Arial (a proportional sans-serif font), and Times New Roman (a proportional serif font).
  - **Size:** Choose a point size. DroidDB initially displays the most commonly used sizes in the drop-down, but you can use the keyboard to enter any size between 4 and 127 points.
  - **Bold, Italic, Underline:** You can apply any or all of these styles to the control.
3. Click **OK** to enter your selections and return to the Properties dialog box.

## Changing Properties for Multiple Controls

Using this feature, you can select two or more controls and set the position, size, font, background color and foreground color properties for all of them at once.

1. While pressing the Control key on the keyboard, select each control. Alternatively, you can left-click the mouse and drag the cursor across the controls.
2. Select **Edit > Property**.

DroidDB displays the Properties dialog box, which contains the following fields:

Field	Description
Left	The horizontal position of the left side of the controls.
Top	The vertical position of the top of the controls.
Width	The width of the controls.
Height	The height of the controls.
Font	The font, size and style of the type displayed by the controls. Click to modify. Refer to the topic, "Specifying Font Properties for a Control," on page 174 for instructions.
Text	The color of the type that appears in the controls.
Back	The color of the background in the controls.

### Note

DroidDB applies any changes you specify in this dialog box to all controls selected in step 1. If you want individual controls to retain their settings for some of the properties, simply leave the initial values in place.

3. Make any changes that you want to apply to ALL selected controls, then click **OK**.

# Arranging Controls on the Form

## Turning Off Snap-to-Grid

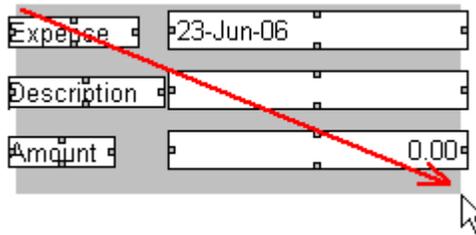
When the "snap-to-grid" feature is on, DroidDB "snaps" controls to an invisible grid as you drag them, helping you align and size controls uniformly. Sometimes, though, you may want to position controls more freely.

- To temporarily turn off the snap-to-grid feature, simply press the Shift key on the keyboard as you drag the control.
- To drag multiple controls this way, hold down the Control key while selecting each one (or left-click and drag across the controls), then hold the Shift key as you drag the controls into position.

## Aligning and Sizing Controls on the Form

You can use the Align command in the Edit menu to precisely align and size selected controls relative to one another.

1. Select all of the controls you wish to align or make the same size using any of the following methods:
  - Select **Edit > Select All**, or
  - While pressing the Control key, click each of the controls, or
  - Left-click the mouse and drag over the controls. The selection area is indicated by a gray box. Release the mouse button. (This method works the same as a Windows marquee select.)



### Note

DroidDB aligns or sizes all of the selected controls to the first control you select. The selection handles around that control have solid boxes, while the others have empty boxes.

2. Select **Edit > Align** and the alignment option you want:

Option	Action
Top	Aligns the top edges of the controls.
Center	Aligns the top/bottom centers of the controls in a horizontal line.
Bottom	Aligns the bottom edges of the controls.
Left	Aligns the left edges of the controls.
Center	Aligns the left/right centers of the controls in a vertical line
Right	Aligns the right edges of the controls.
Width	Makes all selected controls the same width as the first selected.
Height	Makes all selected controls the same height as the first selected.

## Applying Cut, Copy, and Paste to Controls

You can use the Cut, Copy, and Paste commands to duplicate or move one or more controls. The Cut and Copy commands place a copy of the controls on the Clipboard. You can then paste the controls anywhere in the current form or in a different form.

To cut, copy, or paste one or more controls:

1. Select the control(s) you want to cut or copy. If you want to perform the operation on multiple controls, press the Control key on the keyboard while selecting each one, or left-click and drag across the controls.
2. Choose **Edit > Cut** or **Edit > Copy**. Alternatively, you can click the **Cut**  or **Copy**  button.
3. If you want to paste the control(s) into another form, open the form now.
4. Choose **Edit > Paste** or click the **Paste** button .
5. The next step depends upon whether or not you are pasting controls that are connected to table columns:
  - If you are cutting/copying just one control and that control is not connected to a table column or variable, DroidDB just pastes the control into the form.
  - If you are cutting/copying just one control and that control is connected to a table column or variable, DroidDB displays the Select Column dialog box. Click the column to which the control relates, then click **OK**.
  - If you are cutting/copying multiple controls that are connected to table columns and pasting them into the same form, DroidDB automatically pastes the controls and associates them with the same table columns with which they were originally associated.
  - If you are cutting/copying multiple controls that are connected to table columns and pasting them into another form, DroidDB automatically pastes the controls if they have matching table columns (same name and datatype) in the second form's table and ignores any that do not. DroidDB beeps to alert you if it could not paste all of the controls.

## Deleting Controls

To delete one or more existing controls:

1. Click the control to highlight it. If you want to delete multiple controls, hold down the Control key while clicking each one with the mouse pointer, or left-click and drag across the controls.
2. Select **Edit > Delete** or click the **Delete** button .

# Defining Record Sorts and Searches

## Specifying an Initial Sort/Search Ordering

You have the option to specify an initial sort/search order for record display – DroidDB's default is unsorted records. The sort/search order can be based on the value in any column.

The sort/search order determines the records that DroidDB scrolls to when the application's user applies the First, Previous, Next, and Last buttons (or their Record menu equivalents). It also enables the Search function, which allows users to locate records based on the column specified.

To specify a sort/search order:

1. Select **Form > Sort on**.

DroidDB displays the Select Column dialog box.

**Note**

Only "indexed" columns can be used for sorting. If the desired column is not visible in the list, it's probably not indexed. See "Indexing Columns," on page 181 for instructions.

2. Click the **Not Sorted** checkbox to remove the checkmark.

Click the column on which you want sorting based, then click **OK**.

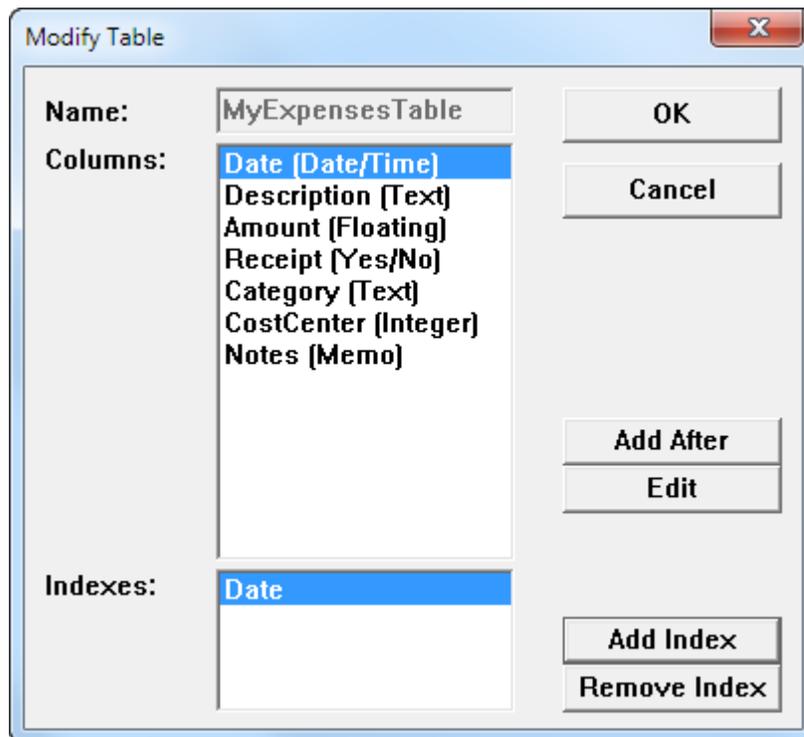
## Indexing Columns

You can index up to four columns per table at one time.

Working in DroidDB's development environment on a desktop PC:

1. Close all forms on the Android device.
2. Open the form that is built on the table you want to modify.
3. Select **File > Modify Table**.

DroidDB displays the Modify Table dialog.



4. DroidDB allows up to four indexes at one time. If there are already four column names listed for "Indexes," and the desired column is not among them, you must remove one to make way for the addition. To do so, highlight one of the column names in the Indexes box, and click **Remove Index**.
5. Click the **Add Index** button.  
DroidDB displays the Select Column dialog box.
6. Highlight the desired column, then click **OK**.

## Creating Pre-defined Filters for Record Display

A filter is a set of conditions that a record must satisfy in order for it to be included in the record display.

Working in DroidDB's development environment, you can create and save any number of filters as part of the form design. You can enable users to apply the filter via a command button, or you can incorporate the filter into a macro.

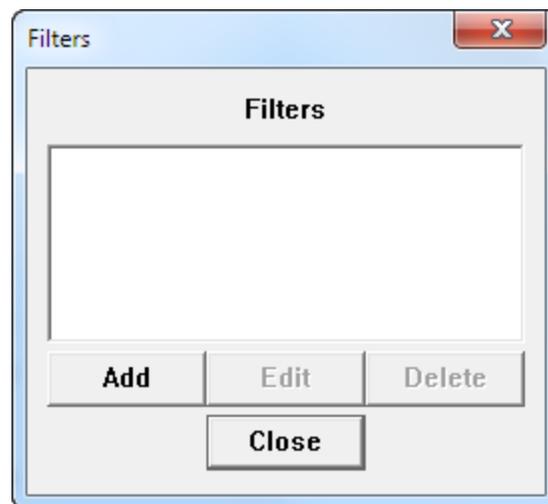
For each filter, you can specify one to three conditions, and you can specify whether the record must match at least one, or all, of the conditions.

Each condition consists of a column name, a value, and a comparison operator that specifies how a record's value in the column must compare to the value in the filter. The filter value may be a constant or a global variable.

To create and save a filter in the DroidDB development environment:

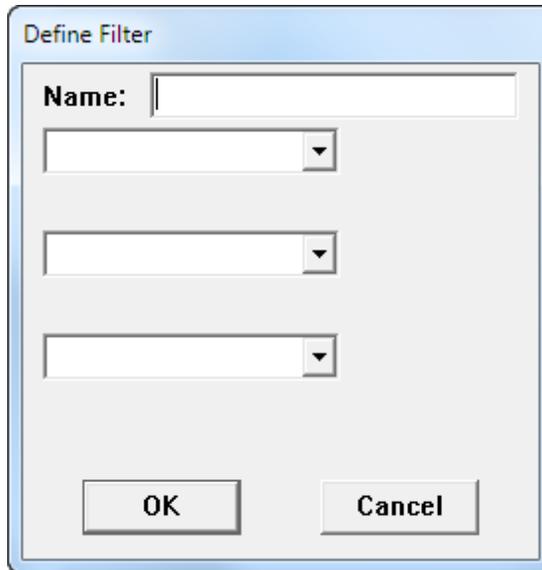
1. Select **Edit > Filters**.

DroidDB displays the Filters dialog box.



2. Click the **Add** button.

DroidDB displays the Define Filter dialog box.



3. In the **Name** field, assign a short but descriptive name to your filter.  
This name will enable you to identify the filter when incorporating it into a command button or macro step.
4. Specify the first condition:
  - Click the drop-down arrow for the first field, and select the column that contains the data you want to compare.
  - In the field that appears to the right of the column name field, select a comparison operator. Not all comparison operators are available for every type of column.

Symbol	Comparison operator
=	Equal to
<>	Not equal to
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
Contains	Contains a specified text string; e.g., contains cook would find "John Cook," "Cooking," "Chocolate cookies," etc.

- Specify the comparison value – either a constant or a global variable. To specify a constant, simply type it in the field. To specify a global variable, click the @ sign to the right of the field and select the variable from the list that appears.

**Note**

Text comparisons are NOT case-sensitive.

**Tip**

You can test for null values (i.e., empty fields). For the comparison operator, select = (Equal to); and for the comparison value, enter "" (two single quote marks with no intervening space).

5. If desired, add one or two more conditions in the remaining fields using the same procedure.
6. If you supplied two or three conditions, choose one of the following:
  - **Match all** A record will be included in the display only if it satisfies **all** of the conditions
  - **Match at least one** A record will be included in the display if it satisfies **any one** of the conditions
7. Click **Close** to close the dialog box.

To make the filter available to your form users, incorporate the “Option | Filter | Predefined” command (page 246) in a command button or macro step. You might also provide a command button or macro step that incorporates the “Option | Filter | Off” command to turn the filter off.

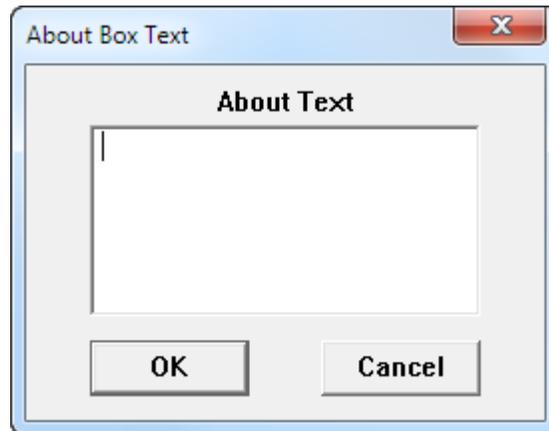
## More Form Customization

### Creating Text for the "About Box"

You can create a brief description of your form that displays when a user selects "About" from the form's Option menu.

1. Select **Form > About Text**.

The About Box text dialog box appears.



2. Enter the text, as you want it to appear in the About Box, and click **OK**.

## Allowing Users to Insert and/or Delete Records

You can specify whether or not users can insert and/or delete records.

### To allow users to insert records:

1. Click **Form** in the development environment's menu bar.

The **Allow Inserts** option "toggles" on and off:

- If it is checked, users will be allowed to add new records to the table.
- If it is unchecked, users will not be allowed to add new records. (They can still modify existing records.)

2. To "toggle" the current setting, click the option.

### To allow users to delete records:

1. Click **Form** in the development environment's menu bar.

The **Allow Deletes** option "toggles" on and off:

- If it is checked, users will be allowed to delete records from the table.
- If it is unchecked, users will not be allowed to delete records. (They can still modify existing records.)

2. To "toggle" the current setting, click the option.

## Enabling the Auto Recalc Feature

The Auto Recalc feature enables you to specify when DroidDB updates values for the calculated field, grid, and lookup controls in a form:

- If this option is enabled, DroidDB automatically recalculates the values whenever the user changes a value in any control on the form.
- If this option is disabled, the user must explicitly select the application's Option > Recalc command to update values in any calculated fields, grids, and lookup controls in the form.

### Note

If your form includes dependent drop-downs, turn Auto Recalc on. Otherwise, the dependent drop-downs may not work properly.

To set Auto Recalc:

1. Click **Form** in the development environment's menu bar.
2. The Auto Recalc feature "toggles" on and off. To change the current setting, click the option.

## Locking the Form Design

You can "lock" a form design to protect it against unauthorized or unwanted modification. This is especially useful if you distribute your DroidDB applications to other users and want to make sure that your design stays unchanged.

### Note

Once you lock a form and save it, no one – including you—can open it in design mode. Be sure to make and store an unlocked copy before saving the locked version, in case you decide to modify the form later.

### To lock a form:

- Select **Form > Lock**.

DroidDB displays a message warning that the lock will be permanent.

## Reports (Business Edition Only)

The DroidDB Business Edition includes a Report Writer. This report writer can generate reports and email them or save them to a file. For details, please refer to the report writer documentation at <http://www.droiddb.com/docs/report.pdf>.

To launch the report designer, click the Windows **Start** button, then **Programs > SYWARE DroidDB > DroidDB Report**.

As with the DroidDB form builder, you will be asked for the name of the application. The File > Open and File > Save commands will put the report templates in the application folder.

Report templates are files with the extension `.drp`. If you have an existing Report CE template and the DroidDB tables have the exact same structure as the Report CE tables, you can copy `MyReportTemplate.rce` to the application folder and rename it `MyReportTemplate.drp`.

# Managing, Testing, and Distributing Applications

## Saving an Application

As you create your application, it's good practice to save your work periodically.

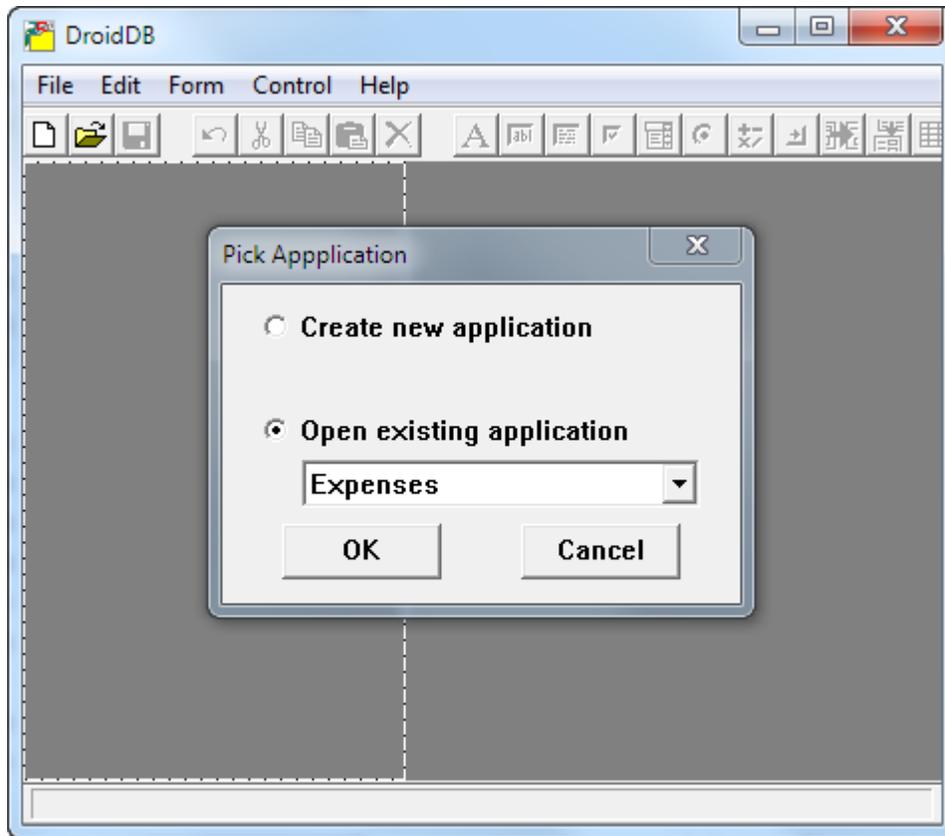
- To save the current form, select **File > Save** or click the **Save** button .
- To save a copy of the current form with a new name, select **File > Save As**. The new form will write to the same table as the original form.

## Opening and Modifying an Existing Application

To open an existing application in DroidDB's development environment on the desktop PC for viewing or modification:

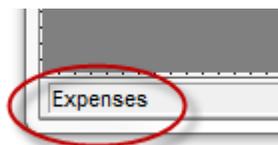
1. If another DroidDB application is open, close the development environment (File > Close).
2. Start the DroidDB development environment.

The Pick Application dialog box opens.



3. Select the desired application and click **OK**.

DroidDB displays the name of the current application in its status bar (lower left).



4. You can now make changes to the application using any of the following commands:

To	Do This
Add a new form.	File > New Form
Open an existing form to view or modify.	File > Open

Add a new form by making a copy of the current form and saving it with a new name. The new form reads/writes data to the same table as the original form.	File > Save As
Delete a form.	File > Delete Form
Add a new table.	File > Create Table
Delete a table.	File > Delete Table
Add a new table and form by downloading a table (and optionally the data it contains) from an ODCB-enabled database on the desktop PC.	File > Download Table
(Business Edition Only) Set up and/or execute synchronization between the application on the Android device and an ODCB-enabled database on the desktop PC or server.	File > Synchronize
(Business Edition Only) Create installation files for distributing the application to other users.	File > Create Distribution Files

5. Once you have opened a form, you can additionally apply these commands:

To	Do This
Save changes to the currently displayed form.	File > Save
Add a new form by making a copy of the current form and saving it with a new name. The new form reads/writes data to the same table as the original form.	File > Save As
Add columns and indexes to the table that the currently open form is built on.	File > Modify Table

## Converting a Visual CE Application to a DroidDB Application

If you have a Visual CE application that was created for Windows CE, you can quickly and easily convert it to a DroidDB application that runs on an Android device.

1. Open the DroidDB development environment on the desktop PC, and select or create an application.
2. Copy the .vce file into the application folder on the Android device.
3. While pressing the Ctrl key, select **File > Open**.
4. In the file selection box, open the .vce file.

DroidDB automatically converts the .vce file to the .ddb file.

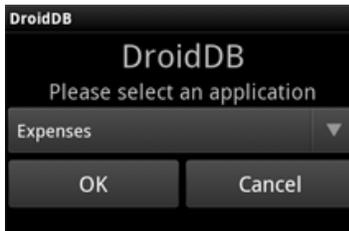
## Testing an Application

Before using the application in "real life," you should test it thoroughly to be sure it works as you intended. You can change your application easily at this point.

### To test an application:

1. Disconnect the Android device from the desktop PC.
2. To start the DroidDB runtime software on the Android device, tap the **DroidDB** icon.

DroidDB asks you to select the application.



3. Choose the desired application. Click **OK**.
4. Create several records to be sure the application works correctly.

If your application works correctly, you are ready to use it. If not, you have probably made a simple mistake or two.

To correct any mistakes, close the application and return to the DroidDB development window. If you made a mistake with a control, change the control's properties or delete and recreate the control. If your mistake concerned the title bar or sort/search order, respecify the option correctly. After making your changes, disconnect the Android device from the desktop PC, open the application and retest.

## **Archiving and Deleting Applications**

DroidDB puts all of the application files on the Android device's storage card. Each application is fully contained in a top-level folder that has the same name as the application. DroidDB does not save any files on the desktop PC. If you want to backup any of the files, simply copy them from the file system on the storage card to a location on your desktop PC or elsewhere.

To delete an application from the Android device, simply delete the application folder from the Android device.

### **Tip**

You'll find a detailed description of the types of files that make up a DroidDB application in the topic, "DroidDB Application Architecture," on page 12.

## Creating Distribution Files (Business Edition Only)

Your DroidDB license allows you to distribute the DroidDB applications that you have created to third-parties.

### Note

In order for end-users to run your applications on their Android device, they must purchase a DroidDB Runtime – End User License from either the DroidDB store (<http://www.droiddb.com/store/>) or the Android App Market.

Using the Distribution Files Wizard that is part of the DroidDB Database & Forms Builder - Business Edition, you can quickly and easily create a complete set of installation files for your users – including not only the primary form and table with indexes, but also any related forms and tables, synchronization settings, embedded graphics, and a setup.exe. You can distribute these files to your users. To install your application on their Android device, all your users have to do is install the runtime software, connect their Android device to their desktop PC, and run setup.exe on the PC.

### Getting Ready

Before creating the distribution files, you must create and thoroughly test the application. Here are some tips and guidelines:

- **Test the application**

Get your application working on the Android device. Test it thoroughly.

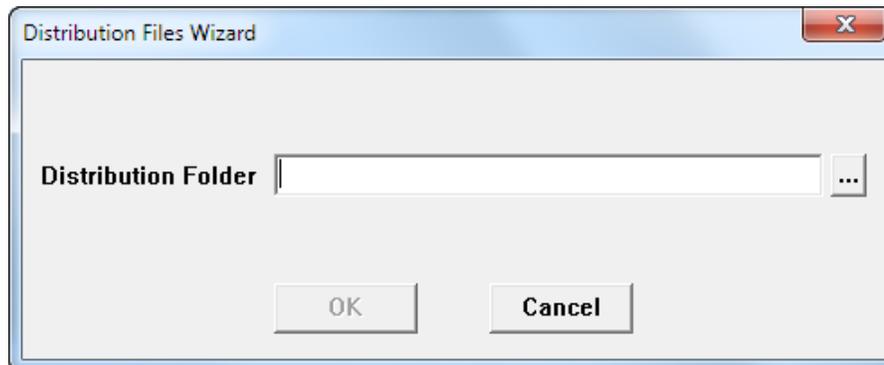
- **Synchronization settings**

The installation program you create will install the synchronizer software and the synchronization settings. If you created the handheld table by downloading it from the desktop PC to a development device, the synchronization settings were specified during that process. You can create, review, and revise the synchronization settings using the DroidDB development environment on your desktop PC (see page 289).

To create an installation disk (or other media) for your users using the Distribution Files Wizard:

1. Connect the Android device to your desktop PC.
2. Open the DroidDB development environment on your desktop PC, and open the application whose files you wish to distribute.
3. Select **File > Create Distribution Files**.

DroidDB displays the Distribution Files Wizard.



4. Specify the target folder on the desktop PC where DroidDB will put the installation files: You can type the path, or click the Browse button  to open the Save As dialog window in which you can navigate to – or create – the target folder. (To create a new folder in the Save As window, click the Browse button, navigate to the place in your file system where you want the new folder, click the New Folder Icon , name the folder, and finally double-click the folder to open it.) Once you have opened the new or existing target folder in the Save As dialog window, click **Save**.
5. Droid DB displays the name of the distribution folder in the Distribution Files Wizard dialog. Click **OK**.
6. After a short time, DroidDB displays the message, “Runtime creation complete.”
7. To create an installation disk for your users, simply copy the contents of the folder that you specified in Step 4 to a CD, disk, or other media.

**To install your application, your users need only:**

1. Install the runtime.
2. Connect the Android device and desktop PC.
3. Insert the CD in the drive (or copy the distribution folder onto the PC hard drive).
4. Run setup.exe.

**Note about Multiple Handhelds Synchronizing to the Same Table**

If you specified “Multiple Handhelds Synchronizing to the Same Table” as part of the synchronization options, each of the distributed DroidDB applications will be able to synchronize with the desktop/server database. However, you must give each handheld a unique name.

The following steps are required for each device (not each application):

1. Attach the Android device to the desktop PC.
2. Open the DroidDB development environment on your desktop PC, and open the application.
3. Select **File > Handheld Name**. Give the device a unique name.
4. Close the application.



## **Part III: Expressions and Functions**

# Using Expressions and Functions

## Introducing Expressions

Expressions are formulas that produce a value. You can use expressions in many places in your forms – in calculated field controls, in macro scripts, and in command buttons. You can use them to perform operations on table column values, global variables, constants, and functions to return numeric, text, or date/time values.

Here are a few examples of expressions:

This Expression	Returns
<code>UnitPrice * Quantity</code>	3.00 when the value in the UnitPrice table column is 1.50 and the value in the Quantity table column is 2
<code>"First Name" &amp; ' ' &amp; "Last Name"</code>	'Jane Smith' when the value in the First Name table column is 'Jane' and the Last Name table column is 'Smith'
<code>@now + (5 * @days)</code>	The date five days from today

### About Constants, Column Names, Functions, and Operators

You build expression by combining constants, column names, functions, global variables, and operators:

#### ■ Constants

A constant is a value that does not change. Constants can be text, numeric, or date/time.

- **Text constants:** Text constants *must always* be enclosed in single quote marks; e.g., 'client'.
- **Date/Time constants:** Date/time constants *must always* be enclosed in single quote marks and use the following formats:

Date: 'yyyy-mm-dd' '2012-02-14' for February 14, 2012

Time: 'hh:mm' '13:41' for 1:41 p.m.

Date/time: 'yyyy-mm-dd hh:mm' '2012-02-14 13:41' for 1:41 p.m. on Feb. 14, 2012

#### ■ Column Names

If you include a reference to a column in the expression, DroidDB uses the value stored in the current record's column. You can refer to columns that store text, numbers, yes/no, or dates/times. If the column name includes blanks or special characters, you must enclose it in double quotes; e.g. "Employees First Name".

#### ■ Functions

Functions are pre-defined, named formulas included with DroidDB. They perform a wide variety of useful calculations. You specify the function by name, and DroidDB performs the calculation.

Functions are identified with a leading @ sign, and use this syntax: @functionName(parameter, parameter...). A parameter can be any expression (including one that contains another function). Some functions require you to specify more than one parameter, in which case you must separate individual parameters with a comma.

DroidDB's most frequently used functions are described in the topic, "Using Basic Functions," on page 204. DroidDB offers other, more advanced functions, which are described in the topic, "Using Advanced Functions," on page 205.

#### ■ **Global Variables**

Global variables make it possible to store intermediate values so that they are available for further processing, without having to write them to a table. The Assign command writes values to global variables, as do many controls. You can access these values in any expression using the @var( ) function.

DroidDB stores values for up to 100 different global variables at one time: @var(0), @var(1) ... @var(99). The variables are global and persistent within the context of a DroidDB application -- meaning that assigned values are available to all forms belonging to an application, and that they persist until overwritten by new values -- even when the application is closed and the device turned off.

For more information about global variables, refer to the topic, "About Using Global Variables to Store Temporary Values", on page 107.

#### ■ **Operators**

Operators are symbols that specify the operation to perform.

##### ■ **Mathematical operators** (use with numeric arguments):

- + addition
- subtraction
- \* multiplication
- / division

If you use more than one operator, DroidDB evaluates the expressions from left to right, and performs multiplication and division before addition and subtraction. You can change the precedence by inserting parentheses: DroidDB performs the operations enclosed in parentheses first. For example, the result of 500 – 300 – 100 is 100. You could insert parentheses to change the expression to 500 – (300 – 100), which returns 300 instead.

##### ■ **Text operator (use with text arguments)**

- & concatenation

It is important to match the operator to the data type of the arguments. DroidDB automatically interprets arguments on either side of a mathematical operator as numbers, and those on either side of a concatenation operator as text. Consider the following expression:

```
(3 + 9) & (1 + 3)
```

It returns '124', a text string.

#### **About IF-THEN-ELSE**

You can also create special expressions that return a different result depending upon a comparison. These "IF-THEN-ELSE" expressions are described in the topic, "Using If-Then-Else in Expressions," on page 203.

## Using Date/Time Values in Expressions

You can use values in Date/Time columns or variables as arguments in an expression. This is especially useful for determining duration (the number of minutes that elapsed between two date/times) or to find a date that is some number of days before or after a given date.

For example, if your billing rate were \$50.00 per hour, the following expression would calculate the charge for the time spent on a job, assuming that you stored the start- and end-times for jobs in columns named `START_TIME` and `END_TIME`:

$$((\text{END\_TIME} - \text{START\_TIME}) / 60) * 50$$

Note: If you use this example in a calculated field, be sure to click the "Money" checkbox when specifying the calculated field control's properties so that DroidDB will display the results with two decimal places.

When building an expression that refers to date/time values, you can use the same kinds of references that you use for other data types; that is, date/time constants, the names of table columns or variables that contain date/time values, and functions that return date/time values (such as `@now`).

- **Date/Time Constants and Column Names:** For instructions on formatting date/time constants and column names, refer to the section, "About Constants, Column Names, Functions, and Operators," in the topic, "Introducing Expressions," on page 200.
- **Functions That Return Date/Time Values:** DroidDB provides many functions for working with dates/times. Three of the most useful are `@now`, `@today`, and `@days`:
  - To use the current date/time in an expression, use the function `@now`.
  - To use the current date in an expression, use the function `@today`.
  - To find a date some number of days before or after a given date, use the function `@days`.

For details about `@now`, `@today` and `@days`, refer to the topic, "Using Basic Functions," on page 204. For information about all other date/time functions, refer to the topic, "Using Advanced Functions - Date/Time Functions," on page 207.

### Note

The internal representation for date/time values in DroidDB is the number of minutes from an arbitrary point in time, established as part of the software design. You can therefore add, multiply, and divide these values, but it is really only meaningful to subtract one value from another to determine the duration.

## Using If-Then-Else in Expressions

You can create an expression in which DroidDB applies a different action depending upon whether a condition is met, such as whether or not the current record matches specified criteria.

For instance, say you want to apply a 5% discount to sales orders over \$500. You could use the following If-Then-Else expression to compute the discount:

```
IF SUBTOTAL > 500 THEN SUBTOTAL * 5/100 ELSE 0
```

### Syntax

Create If-Then-Else expressions using the following syntax:

```
If <boolean-expression> then <expression1> else <expression2>
```

Where:

- <boolean-expression> is the condition that DroidDB uses to evaluate the current record. This expression compares the result of two expressions using any of the following relational operators:

```
<    less than
<=   less than or equal to
=     equals
>     greater than
>=   greater than or equal to
<>   not equal to
```

You can use the logical connectors AND, OR, and NOT to join nested boolean expressions.

- <expression1> Specify the expression that DroidDB applies when the boolean expression is true for the current record.
- <expression2> Specify the expression that DroidDB applies when the boolean expression is false for the current record.

### Tip - Testing for empty record fields:

You can test whether or not the current record contains a null value (is empty) by comparing the table column to an empty string. This works for columns of any data type.

The syntax of the boolean expression must be:

```
<column_name> = ''
```

That is, the name of the table column, equals sign, then two single quote marks without an intervening space.

Example:

```
date = ''
```

## Using Basic Functions

DroidDB provides basic functions that you can use in expressions: @now, @today, @days, @count, and @devicename. In addition to these basic functions which are described below, DroidDB offers many advanced functions which are described in the following topic, "Using Advanced Functions," on page 205.

Function	Returns...
@now	Current date/time. Useful for calculating the elapsed time between a date/time of interest and now.
@today	Current date, without time. Useful for displaying the current date or calculating the number of elapsed days between a date of interest and today.
@days	Number of minutes in a day (24 * 60). Useful for finding a date some number of days before or after a given date.
@count	Number of records currently stored in the entire table or a filtered list.
@devicename	Name of the Android device. Useful for audit trails.

### Examples:

- To find the due date of a bill:

```
"Invoice Date" + (30 * @days)
```

If the invoice date stored in the record was Nov 18, 2012, the result would be Dec 18, 2012.

- To find someone's age in years (assuming that your application stores their birth date in a column named "birthday"):

```
(@today - birthday) / (365.25 * @days)
```

- To create a timestamp:

```
'This record created on ' & @now
```

**Sample result:** This record created on 8/23/2011 10:47:49AM

## Using Advanced Functions

DroidDB offers a large collection of advanced functions that perform a variety of useful calculations. They fall into the following categories, according to the type of data on which they operate: **Text**, **Numeric**, **Date/Time**, **Global Variables**, **GPS**, and **System**. (In addition to the specialized functions described in this topic, DroidDB offers basic functions which are described in the previous topic, "Using Basic Functions.")

Here are a few examples of advanced functions:

This function	Returns
<code>@repeat('abc', 3)</code>	'abcabcabc'
<code>@sqrt(144)</code>	12
<code>@year(Birthday)</code>	1980 when the value stored in Birthday is 1/1/1980

### Text Functions

Text functions manipulate text — they count, combine, search for, insert, replace, and convert character strings.

Function	Description
<code>@ascii(text)</code>	Returns the ASCII code value (integer) of the leftmost character in <i>text</i> .
<code>@char(ASCII_code_value)</code>	Returns the character that has the specified ASCII code value. The value must be between 0 and 255, otherwise, the return value is undefined.
<code>@insert(text, start, length, insert_text)</code>	Returns a character string, where number of characters specified by <i>length</i> have been deleted from <i>text</i> , beginning with the character position indicated by the value of <i>start</i> , and where <i>insert_text</i> has been inserted into <i>text</i> , beginning at <i>start</i> .  For example, <code>@insert('ABCDF', 3, 2, 'XYZ')</code> returns 'ABXYZF'
<code>@lcase(text)</code>	Returns <i>text</i> as all lowercase.
<code>@left(text, count)</code>	Returns the leftmost <i>count</i> characters in <i>text</i> .
<code>@length(text)</code>	Returns the number of characters in <i>text</i> , excluding trailing blanks.
<code>@locate(search_string, text, start)</code>	Returns the starting position of the first occurrence of <i>search_string</i> within <i>text</i> . The search begins with the character position indicated by the value of <i>start</i> (1 indicates the first character position). If the search string is not found, the function returns 0.

<code>@ltrim(text)</code>	Returns the characters in <i>text</i> , without leading blanks.
<code>@repeat(text, count)</code>	Returns the characters in <i>text</i> , repeated <i>count</i> times.
<code>@replace(text, search_string, replacement_text)</code>	Search <i>text</i> for <i>search_string</i> , and replace with <i>replacement_text</i> in all occurrences.
<code>@right(text, count)</code>	Returns the rightmost <i>count</i> characters in <i>text</i> .
<code>@rtrim(text)</code>	Returns the characters in <i>text</i> , without trailing blanks.
<code>@space(count)</code>	Returns a character string consisting of <i>count</i> blank spaces.
<code>@substring(text, start, length)</code>	Returns characters in <i>text</i> , beginning at the character position specified by <i>start</i> (1 indicates the first character position), through the number of characters specified by <i>length</i> .
<code>@ucase(text)</code>	Returns <i>text</i> as all uppercase.

### Numeric Functions

Number functions manipulate numeric data.

Function	Description
<code>@abs(number)</code>	Returns the absolute value of <i>number</i> .
<code>@acos(number)</code>	Returns the arccosine of <i>number</i> as an angle, expressed in radians.
<code>@asin(number)</code>	Returns the arcsine of <i>number</i> as an angle, expressed in radians.
<code>@atan(number)</code>	Returns the arctangent of <i>number</i> as an angle, expressed in radians.
<code>@atan2(number1, number2)</code>	Returns the arctangent of the x and y coordinates, specified by <i>number1</i> and <i>number2</i> , respectively, as an angle, expressed in radians.
<code>@ceiling(number)</code>	Returns the smallest integer greater than or equal to <i>number</i> .
<code>@cos(number)</code>	Returns the cosine of <i>number</i> , where <i>number</i> is an angle expressed in radians.
<code>@cot(number)</code>	Returns the cotangent of <i>number</i> , where <i>number</i> is an angle expressed in radians.
<code>@degrees(number)</code>	Returns the number of degrees converted from <i>number</i> radians. This is especially useful for converting the results of other functions to degrees.

<code>@exp(number)</code>	Returns the exponential value of <i>number</i> .
<code>@floor(number)</code>	Returns the largest integer less than or equal to <i>number</i> .
<code>@log(number)</code>	Returns the natural logarithm of <i>number</i> .
<code>@log10(number)</code>	Returns the base 10 logarithm of <i>number</i> .
<code>@mod(number, divisor)</code>	Returns the remainder (modulus) of <i>number</i> divided by <i>divisor</i> .
<code>@pi</code>	Returns the constant value of pi as a floating-point value.
<code>@power(number1, number2)</code>	Returns the value of <i>number1</i> to the power of <i>number2</i> .
<code>@radians(number)</code>	Returns the number of radians converted from <i>number</i> degrees.
<code>@rand</code>	Returns a random floating-point value.
<code>@round(number, precision)</code>	Returns <i>number</i> rounded to the number of decimal places in <i>precision</i> . If <i>precision</i> is negative, <i>number</i> is rounded to <code>@abs(precision)</code> places to the left of the decimal point; e.g., <code>-1</code> rounds <i>number</i> to the nearest ten, <code>-2</code> to the nearest 100, and so on.
<code>@sign(number)</code>	Returns an indicator of the sign of <i>number</i> : <code>-1</code> when <i>number</i> is negative; <code>0</code> when <i>number</i> equals zero; and <code>1</code> when <i>number</i> is positive.
<code>@sin(number)</code>	Returns the sine of <i>number</i> , where <i>number</i> is an angle expressed in radians.
<code>@sqrt(number)</code>	Returns the square root of <i>number</i> .
<code>@tan(number)</code>	Returns the tangent of <i>number</i> , where <i>number</i> is an angle expressed in radians.
<code>@truncate(number, precision)</code>	Returns <i>number</i> truncated to the number of decimal places in <i>precision</i> . If <i>precision</i> is negative, <i>number</i> is truncated to <code>@abs(precision)</code> places left of the decimal point; for example, <code>@truncate(12654.6, -3)</code> returns <code>12000</code> .

### Date/Time Functions

Date/Time functions manipulate and calculate dates and times.

<b>Function</b>	<b>Description</b>
<code>@dayname(datetime)</code>	Returns the name of the day for <i>datetime</i> . For example, <code>@dayname('2002-02-14')</code> returns Thursday.

<code>@dayofmonth(<i>datetime</i>)</code>	Returns the day of the month for <i>datetime</i> as a integer in the range 1-31.
<code>@dayofweek(<i>datetime</i>)</code>	Returns the day of the week for <i>datetime</i> as an integer in the range 1-7, where 1 represents Sunday.
<code>@dayofyear(<i>datetime</i>)</code>	Returns the day of the year for <i>datetime</i> as an integer in the range 1-366 For example, <code>@dayofyear('2012-02-02')</code> returns 33.
<code>@hour(<i>datetime</i>)</code>	Returns the hour in <i>datetime</i> as an integer in the range 0-23.
<code>@minute(<i>datetime</i>)</code>	Returns the minute in <i>datetime</i> as an integer in the range 0-59.
<code>@month(<i>datetime</i>)</code>	Returns the month in <i>datetime</i> as an integer in the range 1-12.
<code>@monthname(<i>datetime</i>)</code>	Returns the name of the month for <i>datetime</i> . For example, <code>@monthname('2012-02-14')</code> returns February.
<code>@quarter(<i>datetime</i>)</code>	Returns the quarter for <i>datetime</i> as an integer value in the range 1-4, where 1 represents January 1 through March 31.
<code>@second(<i>datetime</i>)</code>	Returns the second in <i>datetime</i> as an integer in the range 0-59.
<code>@year(<i>datetime</i>)</code>	Returns the year in <i>datetime</i> .

### Global Variable Function

Function	Description
<code>@var(<i>n</i>)</code>	<p>Returns the value that has been assigned to the variable. The value can be a text string, date/time, or number.</p> <p><i>n</i> is the identifier of the variable, which must be either a constant between 0 and 99 or an expression that returns a value between 0 and 99.</p> <p>DroidDB stores values for up to 100 different global variables at one time: <code>@var(0)</code>, <code>@var(1)</code> ... <code>@var(99)</code>. The variables are global and persistent within the context of a DroidDB application --meaning that assigned values are available to all forms belonging to an application, and that they persist until overwritten by new values -- even when the application is closed and the device turned off.</p> <p>For more information, refer to the topic, "About Using Global Variables to Store Temporary Values," on page 107.</p>

## GPS Functions

GPS functions return specified data from GPS (Global Positioning System) sentences. These functions are typically applied to text strings returned by a Communicate | GPS command button.

Function	Description
@alt(gpssentence)	Returns the value for altitude from <i>gpssentence</i> *
@lat(gpssentence)	Returns the value for latitude from <i>gpssentence</i> * in decimal degrees.
@long(gpssentence)	Returns the value for longitude from the <i>gpssentence</i> * in decimal degrees.
	<i>*gpssentence</i> is either the name of a table column that holds the GPS data or a global variable that has been assigned GPS data. For example, @alt(gps) or @alt(@var(0))

## System Functions

System functions return values indicating the status of some aspect of the system.

Function	Description
@backupbattery	Returns an integer that indicates how much charge the backup battery has (0 to 100 percent).
@battery	Returns an integer that indicates how much charge the battery has (0 to 100 percent).
@currenttab	Returns index of currently selected tab (0 for first tab, 1 for second tab, 2 for third tab, etc.); returns -1 if there are no tabs on the form.
@fileexists(file_pathname)	Searches the Android device for file_pathname. Returns non-zero value if file is found; returns 0 if file is not found.
@landscape	Returns a value indicating the current orientation of the device window: 0 for No (not in landscape therefore in portrait), 1 for Yes (in landscape), or 2 for square.
@oid	Returns the OID of the current record.
@serialnumber	Returns the serial number of the Android device.



## **Part IV: Macros and Events**

# Macros and Events

## Overview of Part IV

If you took Quick Tour IV, you are familiar with the basics of how macros work and how to create them. This section contains detailed reference information about the DroidDB Macro Builder, macro commands, events, and other aspects of creating and applying macros.

The following topics are discussed:

- **Macros and Events: An Overview**
  - ◇ Introducing Macros and Events
  - ◇ A Sample Macro Script
  - ◇ Steps in Creating a Macro
- **Using the Macro Builder**
  - ◇ Starting the Macro Builder
  - ◇ Creating a New Macro
  - ◇ Editing or Deleting a Macro
- **Creating a Way for Users to Run the Macro**
  - ◇ Command Buttons
- **Triggering Macros with Events**
  - ◇ About Events
  - ◇ Control Events
  - ◇ Form Events
- **Macro Command Reference**
  - ◇ Commands Organized by Category
  - ◇ Commands Organized Alphabetically

# Macros and Events: An Overview

## Introducing Macros and Events

### What is a Macro?

A macro is a script containing a sequence of commands that can be initiated with a single command button or event.

Macros are useful for customizing your forms in any number of ways. For instance, you can save time and effort for your users by defining a series of steps that execute automatically with a single click. You can use a macro to automatically validate user input before it is entered in the table. You can automate complex surveys with many branches using conditional (if-then-else) logic. You can even make your forms dynamic by having macros execute automatically when a given event occurs, such as when the form opens or the user changes a field value.

### Examples of Macros

Quick Tour IV at the beginning of this guide showed you how to create a simple macro that dynamically modifies controls on the form according to user input (specifically, the user's selection in a drop-down list determines what checkbox options are displayed on the form).

Here are just a few more examples to illustrate what can be accomplished with macros:

- You could create a "Submit" macro that, when the user clicks a Submit button, automatically saves the current record with a timestamp and the name of the device that created it. The macro would consist of three commands:

Step 1: An "Assign" command writes the name of the Android device to a column in the table using the @devicename function.

Step 2: An "Assign" command writes the current date and time to a column in the table using the @now function.

Step 3: A Record | Save command saves the entire record to the table.

- You could build a questionnaire in which the questions branch in different directions based on the user's responses. For each branch, you could have a macro consisting of two commands:

Step 1: An "Assign" command stores a value based on the user's input.

Step 2: A "Select Tab" command moves to the tabbed page of the form that contains the appropriate set of questions based on that response.

- You could apply a validation test to user input (for example, age must be greater than 0). If the response is valid, the macro saves the record; if the response is invalid, it displays an error message. This macro could consist of four commands:

Step 1: A "Skip" command evaluates the user input: If age is greater than 0, it skips two commands in the sequence; otherwise, it skips zero commands in the sequence.

Step 2: A "Message" command displays an error message, "Age must be greater than 0."

Step 3: A "Return from macro" command stops the macro.

Step 4: A Record | Save command saves the record to the table.

### **What are the Components of a Macro?**

A macro consists of a sequence of commands that you want performed in a given order.

DroidDB offers a large library of commands that you can use as steps in a macro. Each command type is explained in the command reference section beginning on page 230.

Most commands have parameters you must specify, such as values read into or written out by the command. For these values, you can supply a constant or an expression. Expressions can read values stored in a table column for the current record or in a global variable, as well as apply mathematical operations, if-then-else logic, and functions to return a desired value. Refer to page 200 for details about expressions.

You can also use global variables to pass temporary values between steps in a macro. Refer to page 107 for more information about global variables.

The script commands can execute in a straight linear sequence, or you can incorporate skips, branches and loops based on conditional logic using Skip commands. You can also have submacros within a macro. Submacros are a sequence of commands that you write once, and can then call from the main script as many times as needed. Simply write the submacro as you would any other, then call it from the main script using a "Run macro" command.

One DroidDB form can have any number of macros. However, each macro is "built into" a single DroidDB form; it is not possible to call an external macro.

### **How are Macros Run?**

Macros can be run in a DroidDB application in either of two ways: they can be run when the user explicitly selects a command button that you have associated with the macro, or they can be run automatically when an event occurs (see below).

To associate a macro with a command button, refer to page 226.

To associate a macro with an event, refer to page 227.

### **What is an Event?**

Events are generally (but not always) actions initiated by the user. DroidDB recognizes two categories of events: form events and control events. Form events include opening the form. Control events include "get focus" (the user clicks or tabs to a control), "Lose focus" (the user clicks or tabs away from the control) and "changed" (the user adds, edits or deletes a value in the control).

As explained above, you can associate a macro with an event, so that when the event occurs, the macro runs automatically.

Events are explained in detail on page 227.

### **How Do You Create a Macro?**

The DroidDB Macro Builder is a structured development environment that has been designed to help guide you through the process of creating a macro. For example, to define each step in your macro, you simply choose from the drop-down list of commands, then complete a few input fields to set any additional properties of the command. You use the Macro Builder to arrange the commands in the order you want them to execute.

The Macro Builder also enables you to assign the macro to a form event.

The Macro Builder is described in detail beginning on page 218.

### **Putting it All Together**

The following page provides a sample macro script that illustrates many of the techniques you can use in your own macros. In addition, be sure to check out the SYWARE website regularly, in particular the Tip of the Month section, <http://www.droiddb.com/totm/>, for more sample macros and ideas.

## A Sample Macro Script

The following macro script sets the value in the Amount column to zero for all records in a table. It demonstrates the use of global variables to pass temporary values from one step to the next, as well as the use of Skip commands with If-Then-Else logic that loop through the statements until all records have been processed.

Step	Command	Command parameters	Comment
1	Record   First		Go to first record in the table
2	Assign	Value: @count Col: @var(0)	Count the number of records in the table using @count and write the result to the global variable @var(0)
3	Macro step label	Label: StartOfLoop	
4	Skip	Num: If @var(0) = 0 then go to Done	If no more records to do, skip ahead to the step labeled Done; otherwise, continue
5	Assign	Value: 0 Col: Amount	Write the value 0 to the Amount column for the current record
6	Assign	Value: @var(0) - 1 Col: @var(0)	Subtract 1 from the record count and write the result to @var(0)
7	Record   Next		Go to the next record in the table
8	Skip	Num: Go to StartOfLoop	Go back to the step labeled StartOfLoop.
9	Macro Step Label	Label: Done	
10	Message box	Msg: 'All amounts set to zero'	Display message for user.
11	Return from macro		Return processing control to the form.

## Steps in Creating a Macro

The sequence for creating a macro is:

- Plan the macro
- Create the DroidDB application that will run it.
- Write the macro using the Macro Builder
- Provide a way for the macro to run
- Test the macro

### Plan the Macro

You may find it helpful to plan your macro before writing it. For example:

- What task(s) do you want your macro to perform?
- If the macro processes table records, should the records be sorted as part of the macro?
- Are there sequences of steps that repeat? If so, can you put them in a submacro?
- How will your macro be launched? Will it be tied to a command button that the user clicks, or should it run automatically when a form or control event happens?

Once you've decided what the macro should do, it's a good idea to write out each of the steps in English or "pseudocode" before actually writing the commands in the Macro Builder.

### Create the DroidDB Forms

Create the form and table as you would any other application. If any of the macro steps will use a sort, you must index the sort column first. Similarly, if the macro calls any external forms, it's a good idea to verify that they exist.

### Write the Macro Using the Macro Builder

With the application open in the DroidDB development environment on the PC, open the Macro Builder (explained on page 218). The Macro Builder provides a helpful framework for writing and organizing the commands that make up the steps in the macro.

### Provide a Way for the Macro to Run

You can have the macro run in any of the following ways:

- Create a command button that the user selects (see page 226).
- Assign a form event to trigger the macro automatically (see page 229)
- Assign a control event to trigger the macro automatically (see page 228)

### Save and Test the Macro

Macros are saved when you save the DroidDB application to which they belong.

Test the application thoroughly before using it in production.

# Using the Macro Builder

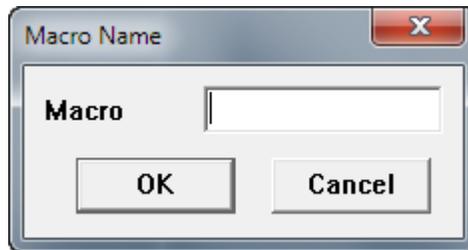
## Starting the Macro Builder

The Macro Builder is a development environment inside the larger DroidDB development environment that provides a helpful framework for creating, editing, and managing macros and associating them with form events.

To start the Macro Builder:

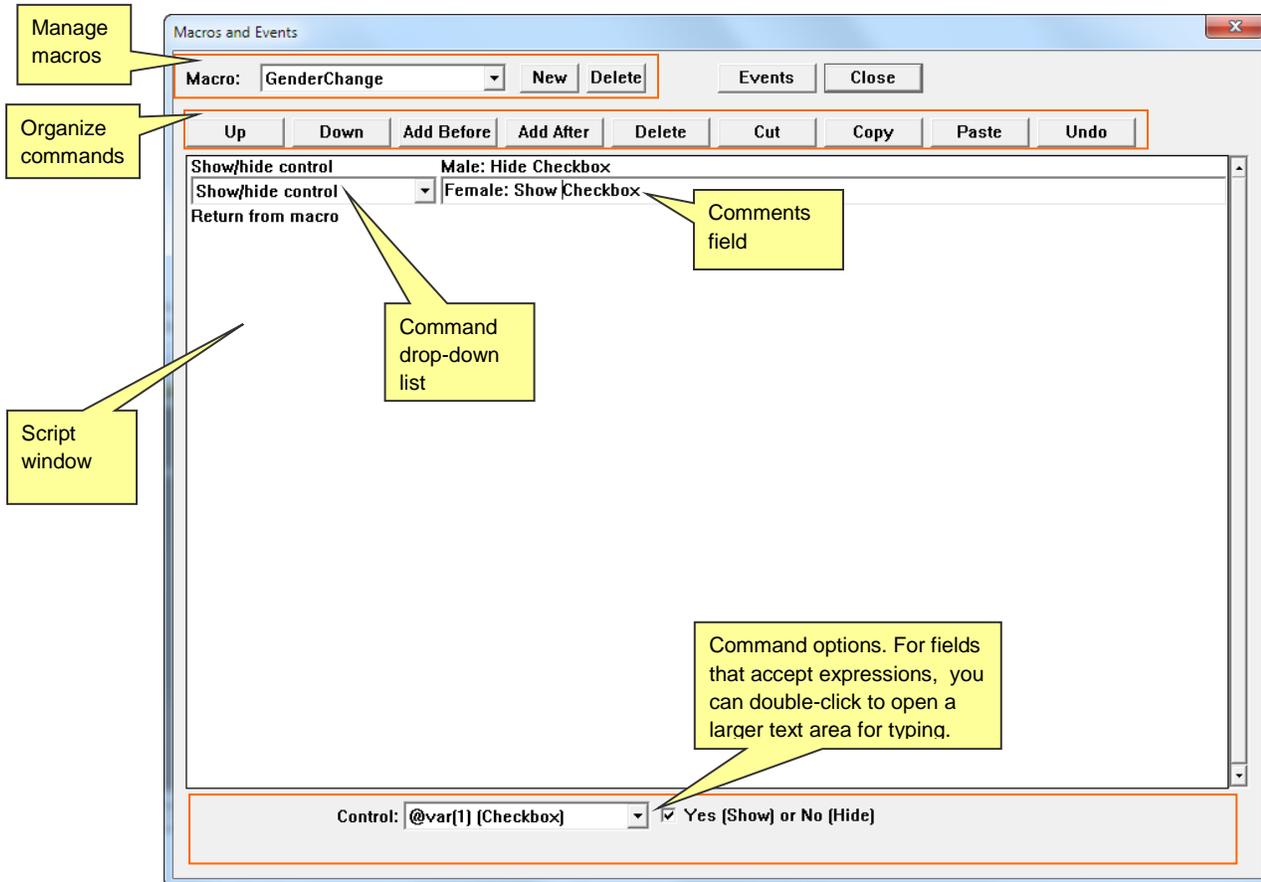
1. Working in the DroidDB development environment on the desktop PC, open the DroidDB application and form in which you want to include the macro.
2. Select **Edit > Macros/Events**.

If you have previously created a macro as part of this application, DroidDB opens the Macro Builder. Otherwise, DroidDB displays the Macro Name dialog box.



3. Enter a meaningful name for your macro that you will be able to recognize later, and click **OK**.

DroidDB opens the Macro Builder. Here you can build your macro scripts and associate them with form events.



The *Manage macros* area enables you to add a new macro, or select an existing macro to modify or delete:

- The **Macro** drop-down list contains an alphabetical list of all macros that you've created in the current DroidDB application so far.
- The **New** button lets you begin the process of creating a new macro.
- The **Delete** button deletes the macro currently displayed in the Macro drop-down list.

The *Organize commands* area provides buttons you can apply to the currently selected command in the Script window:

- The **Up** button moves the currently selected command up one step in the script.
- The **Down** button moves the currently selected command down one step.
- The **Add Before** button adds a new command to the script, just before the currently selected command.
- The **Add After** button adds a new command to the script, just after the currently selected command.
- The **Delete** button removes the currently selected command from the script.
- The **Cut** button removes the currently selected command(s) from the current location so that you can paste them elsewhere in the script.

- The **Copy** button copies the currently selected command(s) so that you can paste them elsewhere in the script or in another macro.
- The **Paste** button inserts cut or copied command(s) just above the currently selected command.
- The **Undo** button undoes all modifications you've made to a command since you last selected it. If the command is new, Undo effectively deletes it.

The *Script window* lists all of the commands that make up the steps in your macro script.

When you click any command line in the Script window, the Macro Builder displays the following:

- The **Command drop-down list** enables you to choose one of DroidDB's pre-defined commands.
- The **Comments field** lets you enter a description of your command.
- The **Command options** area displays fields for entering parameters specific to the command selected in the Command drop-down list. For fields that accept expressions, you can double-click the field to open a larger text area in which to more easily add or edit long expressions.

The **Events** button opens the Events window, in which you can associate any existing macro with a form event. (To associate a macro with a control event, set the Events property of that control.)

The **Close** button closes the Macro Builder window and *temporarily* saves the current version of the macro. *You must save the DroidDB form to save the macro.*

The following topics explain how to create, edit, and delete macros using the Macro Builder.

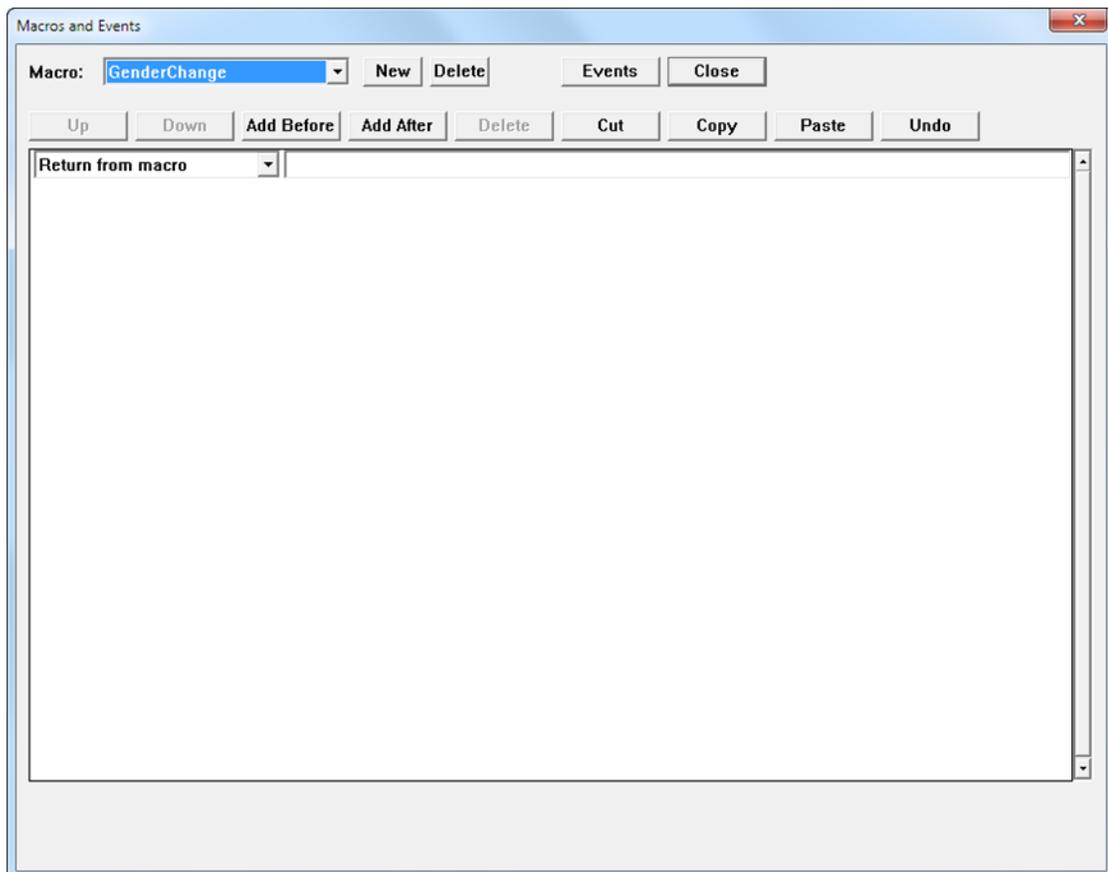
## Creating a New Macro

The Macro Builder makes it easy to create a macro.

To create a new macro:

1. Open the Macro Builder using the instructions in the previous topic.
2. Unless you named your new macro when first opening the Macro Builder, click the **New** button and name your macro now.

The name of the macro you are creating should be visible in the Macro drop-down list at the top-left of the Macro Builder window.



3. Click **Add Before**.

The Macro Builder adds a new command to the top of the Script Window – above the default Return from Macro statement. It is considered good practice that the last statement in your script be a Return from Macro statement, so you should leave it in place.

4. To add a new command to the script:
  - Open the Command drop-down list and select the desired command type. The list is extensive; you may have to scroll to find the command you want.
  - Once you select a command in the drop-down, DroidDB may display zero or more input fields at the bottom of the Macro Builder window, depending upon the command. Complete these

fields to specify the parameters specific to the selected command type. You can double-click an input field to open a larger area in which to type.

**Tip**

All of the available command types, and their optional parameters, are described in detail in the section, “Command Reference” beginning on page 230.

- Enter comments in the field to the right of the command. These comments can be very helpful if you need to modify or debug your script in the future. Also, the Script Window displays only the command name, not the specific parameters, so it can be difficult to tell one command from another without either comments or selecting each one to view its parameters.
5. Once you have created each one of the commands in your script, put them in the order you want them to execute. To move a command up or down in the sequence, select it and click the **Up** or **Down** button.

**Tips**

- For more complicated scripts, you may want to change the flow of execution (the sequence in which the commands are executed) to something other than a straight linear sequence. You can incorporate “If-Then-Else” or even “If – Then – Else If – Then – Else” branches and loops using Skip statements.  
  
You can also break out statements by putting them in submacros. This is useful if you have a set of statements that are used repeatedly in the macro, for example. Rather than rewriting the same set over and over, you can write them once, save them as a separate macro in the application, and call them when needed using a Run Macro command in the main macro.
  - You can also cut, copy, and paste commands within the current script or to and from another macro script. Refer to the topic, “Applying Cut, Copy, and Paste to Macro Commands” on page 223.
6. When you have completed your macro, you can close the Macro Builder by clicking the **Close** button.
7. To save the macro, you must save the application. Select **File > Save**.

**Warning!**

If you close the DroidDB development environment without saving the form, your macro will be lost.

## Applying Cut, Copy, and Paste to Macro Commands

You can cut and paste one or more commands within the current script, or to and from another macro script.

To cut/copy/paste in the Macro Editor:

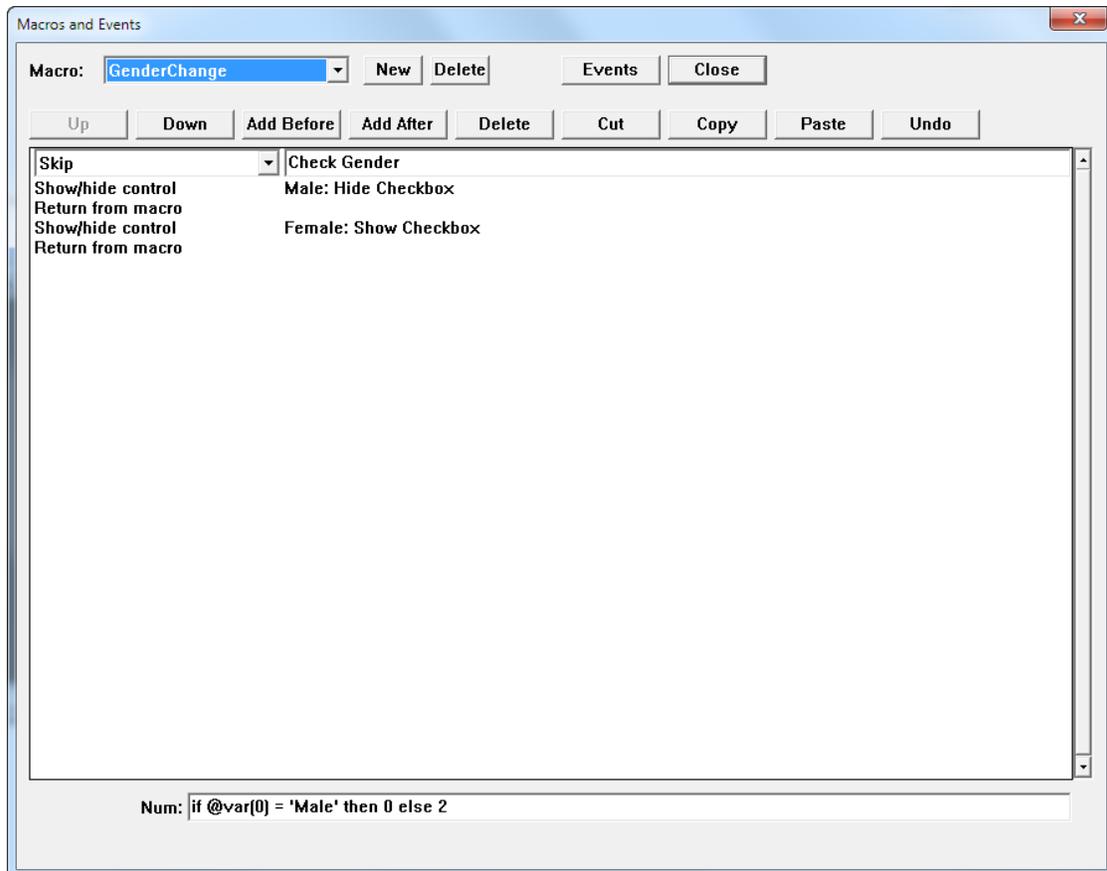
1. Select the commands you want to cut or copy. If you want to select multiple commands, click the first command to highlight it. Then, while holding down the Shift key, click the last command. Or, you can simply drag the mouse pointer over the desired commands (this works best if you start with the last command and drag upwards). If you change your mind and want to deselect the commands, click the Undo button.
2. Click the **Cut** or **Copy** button.
3. If you want to insert the command(s) in another macro, open that now.
4. Select the command that you want to be immediately after the commands pasted in.
5. Select the **Paste** button.

## Editing an Existing Macro

To edit an existing macro:

1. Working in the DroidDB development environment on the desktop PC, open the DroidDB application and form that includes the macro you wish to edit.
2. Select **Edit > Macro/Events**.

The Macro Builder opens.



3. From the **Macro** drop-down menu, select the name of the macro you wish to edit.
4. Edit the commands and command parameters in the Script Window using the editing features of the Macro Builder (as described in the topic "Starting the Macro Builder" on page 218).
5. When you have completed your macro, close the Macro Builder by clicking the **Close** button.
6. To save your changes to the macro, you must save the DroidDB form. Select **File > Save**.

### Important!

If you close the DroidDB development environment without saving the form, your edits to the macro will be lost.

## Deleting an Existing Macro

To delete an existing macro:

1. Working in the DroidDB development environment on the desktop PC, open the DroidDB application and form that includes the macro you wish to delete.
2. Select **Edit > Macro/Events**.  
The Macro Builder opens.
3. From the **Macro** drop-down menu, select the name of the macro you wish to delete.
4. Click the **Delete** button at the top of the Macro Builder window.

### **Important!**

You must save the DroidDB form for any changes to macros to take effect. If you close the DroidDB development environment without saving the form, the macro will not be deleted.

# Creating a Way for Users to Run a Macro

## Creating a Run Macro Command Button

You can place a button on the form that users click to run a macro. You must have already created and named the macro as part of the current DroidDB form.

### Tip

Alternatively, you may want the macro to execute automatically when an event occurs. Refer to the following topics.

To create a command button to run a macro:

1. Select **Control > Command Button**, or click the **Command** button , or right-click where you want the control to appear and select **Command**.  
DroidDB displays the Command Button Properties dialog box.
2. Specify the first eight properties as you would for any other command button. Refer to step 1 in "Creating Command Buttons" on page 154, for details.
3. From the command drop-down list, select **Run macro**.
4. For **Macro**, select the name of the macro you want to run when the user clicks the command button.
5. Click **OK**.
6. If necessary, resize and drag the button to the desired position on the form.

# Triggering Macros with Events

## About Events

Using events, you can create forms that respond to the user dynamically. You can associate a macro with an event, so that when the event occurs, the macro runs automatically.

For example, you could set up a validation macro to automatically validate a price in the Amount field before the user leaves the field, or a macro that shows additional controls on the form when the user selects a checkbox, or a macro that displays a splash screen when the application first opens.

There are two categories of events: control events and form events. Control events occur when a control on the form (e.g., radio button, checkbox, etc.) gets or loses focus or is used to change a value. In the current release there is one form event: the form opens.

## Using Control Events to Trigger a Macro

DroidDB recognizes three types of control events:

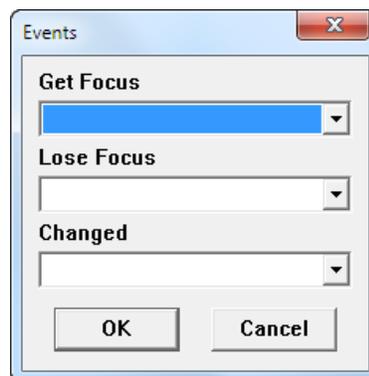
Event	Definition
Get focus	User touches the control, or the Set Focus command puts focus on the control.
Lose focus	User touches away from the control, or the control loses focus because of the Set Focus command.
Changed	User adds, edits, or deletes a value in the control. (Assign commands do not trigger a Changed event.)

You define event responses as properties of individual controls. The table below summarizes the events you can use for each control type:

Control	Events		
	Get focus	Lose focus	Changed
Edit box	X	X	
Note box	X	X	
Checkbox	X	X	X
Radio buttons	X	X	X
Drop down list	X	X	X
Scribble box	X	X	
Image control	X	X	
Grid control			X

To set up control events to trigger macros:

1. In the main DroidDB development environment, open the Properties dialog box for the control that will trigger the macro. Click the Events button. DroidDB displays the Events dialog box.



2. From the drop down menus, select the macro you want to associate with each event for the control. You can assign a macro to any or all of the events, depending upon your design.
3. Click **OK**.

## Using Form Events to Trigger a Macro

The current release recognizes several types of form events :

Event	Definition
On startup	The form opens.
On record display	A record is opened.
On leaving record	User goes to another record. This event could be useful for triggering a validation macro to check that input values are within an acceptable range, for example.
On select tab	The user selects a different tab on the form.

To set up form events to trigger macros:

1. Unless the Macro Builder is already open, select **Edit > Macro/Events**.

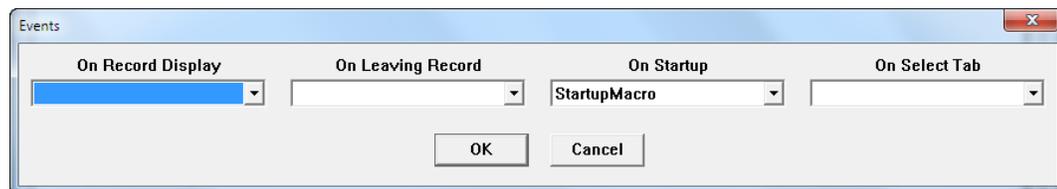
The Macro Builder opens.

### Note

If DroidDB displays the Macro Name dialog box when you first open the Macro Builder, you have not yet created a macro for the current form. Go to the instructions, "Creating a New Macro" on page 221 for instructions.

2. Click the **Events** button near the top of the Macro Builder window.

The Events dialog box opens.



3. Select the macro in the drop-down for the desired form event.
4. Click **OK**.

# Command Reference

This section describes all of the DroidDB commands that you can use as steps in a macro script. Many of the commands can also be assigned to a command button.

The commands are listed two ways – first by category, then in alphabetical order for convenient reference. The alpha listing (page 234) provides a complete description of each command, including parameters you must specify.

## Commands by Category

### Script Control

Use this command	To
Skip <i>num</i>	Skip forward or back in the macro sequence; can be used to create conditional branches and loops. (page 259).
Macro step label <i>label</i>	Identify destination for a Go To clause in a Skip command. (pg 240)
Sleep <i>secs</i>	Freeze the application for some number of seconds. (page 261)
Run macro <i>macro</i>	Start a specified macro. This command makes it possible to incorporate a submacro in the main macro. (page 255)
Timer <i>secs</i> , <i>macro</i>	Start specified macro after delay of some number seconds. (pg 265)
Return from macro	Stop the execution of the current macro and return control to the process that called it. If this statement is included in a submacro, processing returns to the main macro. If included in a main macro, processing returns to the form. Recommended as the last step in every macro. (page 252)
Stop macro	Stop the current macro. If included in a submacro, processing does not return to the main macro. Typically used when an error condition is detected. (page 264)
Message box <i>msg</i> , Yes ( <i>Skip 1</i> ) or No ( <i>Skip 2</i> )	Display a message box and, optionally, execute a different set of macro steps depending upon how the user responds. (page 242)

### Form Controls

Use this command	To
Show/hide control <i>control</i> , Yes ( <i>Show</i> ) or No ( <i>Hide</i> )	Show or hide any control on the form. (page 258)
Set focus <i>control</i>	Put focus on a form control. Frequently used after validation to put focus on a control with failed input. (page 257)
Select tab <i>num</i>	Select a specific tab. (page 256)

**Records**

Use this command	To
Sort by <i>col</i>	Sort records by values in a table column. (page 263)
Record   First	Go to the first record in the table, based on the current sort order.* (page 250)
Record   Previous	Go to the previous record in the table, based on the current sort order.* (page 250)
Record   Next	Go to the next record in the table, based on the current sort order.* (page 250)
Record   Last	Go to the last record in the table, based on the current sort order.* (page 250)
Record   Search <i>exact matches only, arg, skip n if not found</i>	Find the first record (based on the current sort order) whose value in a specified table column matches a search value.* (page 250)
Record   Save	Save the current record.* (page 250)
Record   Insert	Insert a new record.* (page 250)
Record   Duplicate	Create a new record by creating a duplicate copy of the current record. (page 250)
Record   Delete	Delete the current record.* (page 250)
Option   Clear	Delete <i>all records</i> in the table.* (page 244)
Option   Export	Export records from the current table or filtered list to a comma delimited ASCII text file. * (page 245)
Option   Filter   Off	Display all records in the table (page 246)
Option   Filter   Predefined <i>filter, skip n if not found</i>	Run a filter that has been predefined as part of the form design. Note: At least one filter must already be defined as part of the form design in order for this command to be available in the Macro Builder or command button command list (page 246)
Option   Import	Import contents of a comma delimited ASCII text file into the current table.* (page 247)

\*Performs same action as DroidDB menu option of same name.

**Field Values**

Use this command	To
Assign <i>Value Col</i>	Assign a value to a table column or global variable. (page 235)
Option   Recalc	Recalculate values for Calculated Field controls in current record.* (page 248)

\*Performs same action as DroidDB menu option of same name.

**Form Navigation**

Use this command	To
Record   Close	Close the current form.* (page 250)
Option   About	Display information in the About box.* (page 243)

\*Performs same action as DroidDB menu option of same name.

**Data Acquisition**

Use this command	To
Barcode	Activate barcode reader and write scanned input to table column or global variable. (page 236 )
Communicate   GPS	Read \$GPGGA sentences and write them to a table column or global variable; retrieve GPS data in NMEA standard format. Useful for getting and storing latitude, longitude, and altitude. (page 250)

**Run External Programs**

Use this command	To
Jump <i>Search Key</i> , <i>Jump to appl</i> , <i>Only show matches</i> , <i>Record Creation</i>	This command works the same as a Jump button. (page 239)
Run Form <i>form</i> , <i>Close Form</i>	Launch another form in the application. (page 254)
Run External <i>command</i> , <i>arg</i>	Launch other executable program and optionally pass a value from the current record to the second program. (page 253)
Report	Display records in a DroidDB report layout. (page 251)

**Wireless Synchronization**

Use this command	To
mEnable synchronize <i>Quiet</i>	Synchronize all local tables on the Android device with a remote database server. (page 241)

**Miscellaneous**

Use this command	To
Play sound <i>symbol</i>	Play one of five different sounds from a predefined set, or a .wav file. (page 249)
Dial <i>Col</i>	Dial a phone number stored in the current record or global variable. (page 238)
SMS send message <i>Msg, Phon</i>	Send an SMS instant text message to a specified phone number (page 262).
SMS receive message <i>command, arg</i>	Launch a DroidDB form upon receipt of an SMS instant text message that begins with the string <code>.ddb.</code> and optionally save the incoming message to a variable. (page 262)

## Commands Organized Alphabetically

This section lists in alphabetical order all of the commands you can use as steps in a macro program. Many of the commands can also be assigned to a command button.

### **About Expressions**

For many command parameters, you can supply the same type of expression that you can specify in a calculated field control. Expressions can read values stored in a table column for the current record or in a global variable. Refer to page 200 for details about creating expressions.

### **Tip**

When entering or editing expressions in the Macro Builder, you can double-click the field to open a larger text area in which to more easily type long expressions.

## Assign command

### Assign *Value Col*

The Assign command is used in a macro sequence to assign a value to a table column or global variable. It has two parameters:

- **Value:** The value to be assigned. You can supply a constant, or an expression that returns a value (page 200). The data type of Value can be text, numeric or date/time. You can double-click the field to open a larger area in which to type.
- **Col:** The table column or global variable to which the value is assigned.

### Tip

If you wish to create a timestamp, use the Assign command with one of the date/time functions such as @now or @today.

Example: `Assign @now timestamp`

## Barcode command

### Barcode Col

The Barcode command activates a barcode reader and writes the scanned input to a table column or global variable. It has one parameter:

- **Col:** The table column or global variable to receive the scanned input; typically of datatype text.

## Communicate | GPS command

### Communicate | GPS *Col*

You can use this command to retrieve latitude, longitude, and altitude from the Android device's GPS. The GPS data is a string in NMEA standard format.

The command puts the NMEA sentence (starting with \$GPGGA) into the table column or global variable that you specify. It has one parameter:

- **Col:** The table column or global variable to receive the GPS sentence.

#### **Tip:**

You can use the functions @alt, @long and @lat to get the altitude, longitude and latitude values from the sentence. Refer to the section, "GPS Functions," near the end of the topic, "Using Advanced Functions in a Calculated Field," on page 209.

## Dial command

### Dial Col

This command dials a phone number. It has one optional parameter:

- **Col:** The table column or global variable that stores the number to dial. If you do not specify a table column or variable, DroidDB uses the first phone number it finds in the current record.

## Jump command

### **Jump** *Search Key, Jump to appl, Only show matches, Record Creation*

This command works just like having a Jump button (page 161) embedded in your macro sequence. When the command is executed, it takes the search key value for the current record from a table column or global variable, opens the "Jump to" form, and displays the first record in that form's table that stores the same value. The "Jump from" form does not close; tapping the device's Back button after a jump returns to the "Jump from" form.

The Jump command has the following parameters:

- **Search Key** and **Jump to appl**: DroidDB guides you through selecting these first two parameters.

When you select the command in the Macro Builder, DroidDB displays the Select Columns dialog box. Select the table column or global variable in the current form's table that stores the value DroidDB will use to match related records in the "jump to" table. If you select a column, keep in mind that the "jump to" table must have an indexed table column with the same name that stores the same datatype.

Then, DroidDB displays the "Form to Jump to" box. Navigate to the form that will be the destination for the jump – the "jump to" form.

If you initially selected a global variable as the "jump from" key value, DroidDB asks you to select the column that stores the values to match in the "jump to" table. The column you select must be indexed.

After you make these selections, the Macro Builder displays the following options at the bottom of the window:

- **Only Show Matches**: If checked, DroidDB filters records in the "jump to" table so that only records that have a matching value in the key column are retrieved for display. If unchecked, all records in the "jump to" table are retrieved, but the form opens to the first record with a matching value.
- **Record Creation**: Action that DroidDB takes when opening the "jump to" form:
  - ◇ **Always** - Always create a new record. If you select the "Only show matches" checkbox, DroidDB automatically fills in the value from the key column or variable.
  - ◇ **Only if Needed** - Create a new record only if there are no existing records with a matching value in the key column. If you select the "Only show matches" checkbox, DroidDB automatically fills in the value from the key column or variable.
  - ◇ **Never** - Do not create a new record. Display first record in "jumped to" table that has a matching value in the key column. If no matches are found, display messages, "Record not found."

## Macro Step Label command

### Macro Step Label *Label*

This command provides a destination for a Go To clause in a Skip command. It has one parameter:

- **Label:** A sequence of characters (letters, digits, underscores). A label cannot begin with a digit, nor can it include a space. Labels are not case-sensitive.

Example:

```
StartOfLoop
```

**mEnable Synchronize command****mEnable synchronize** *Quiet*

mEnable is a separate SYWARE product that makes it possible for DroidDB users to synchronize local data on their Android devices with a remote server via a wired or wireless connection including the Internet.

This command establishes a connection to the mEnable server and executes full synchronization between all eligible local tables in the current DroidDB application with tables in the server database. It has one parameter:

- **Quiet:** Suppresses the logon box that otherwise appears when a connection is initiated. (When the user initiates the very first mEnable session on the Android device, the logon box appears so that the user can specify the connection settings. mEnable uses those settings for subsequent sessions.) If this option is not selected, the logon box appears every time a new connection is opened.

**Note**

The mEnable server software must be installed and running on the database server. An mEnable client does not have to be installed on the Android device (it is integrated into the DroidDB runtime).

## Message Box command

### **Message box** *msg, Yes (skip 1) or No (skip 0)*

This command displays a message box with custom text that can include values from the current record. Optionally, you can specify that the user must respond Yes or No to the message, and have the macro skip to a different command depending upon how the user responds.

A Message box command has two parameters:

- **msg:** The message to be displayed; can be a literal string or the result of an expression (page 200) that can include values from a table column for the current record or from a global variable. As with any DroidDB expression, you must enclose literal strings in single quotes (') and column names that include spaces in double quotes ("). You can double-click the field to open a larger area in which to type.
- **Yes (skip 1) or No (skip 0):** If you select this option, the message box displays Yes and No buttons from which the user must choose. If the user clicks "Yes," the macro skips one command in the macro sequence; if the user clicks "No," the macro skips no commands in the macro sequence. If you leave this option unchecked, the message box simply displays an OK button that the user clicks to acknowledge and close the box.

**Option | About command**

This command opens the About Box to display any information you have provided there. It works the same as the DroidDB menu option of the same name.

An Option | About command has no parameters.

### Option | Clear command

This command deletes all records in the table associated with the current form. It works the same as the DroidDB menu option of the same name.

An Option | Clear command has no parameters.

#### Note

The purpose of this command is NOT to clear the current record. ***This command deletes ALL records in the current table.***

**Option | Export command****Option | Export**

This command exports records from the current table to a comma-delimited ASCII text file.

It has the following parameters:

- **No headers:** If you select this option, the table column names will not be included in the export file. If you don't select this option, the first line of the export file will consist of the table column names.
- **File:** The table column or global variable that stores the file pathname to export to.

If you do not provide any value for File, DroidDB automatically assigns the name of the table to the export file and puts it in the application's folder on the Android device. For example if the current form in the application MyApp uses MyTable, DroidDB will create a file named MyTable.txt in the folder named MyApp.

If the file to export to already exists, it will be overwritten. You can avoid this by checking with the @fileexists function and notifying the user accordingly.

**Notes:**

- Field values are exported in the same order as the columns in the table.
- All date/time values are automatically exported in the following format:  
YYYY-MM-DD HH:MM:SS
- After the table is exported, you may want to delete all records in the table. (Be sure to keep the archived records in a safe place.) If you don't delete the records, and you later import the file back in, you will have two of every record.

## Option | Filter commands

Filters limit record displays to records that match desired criteria. These commands turn filters off and on.

### Option | Filter | Predefined *filter, skip n if not found*

Runs a filter that has been predefined as part of the form design. At least one filter must already be defined as part of the form design in order for this command to be available in the Macro Builder (or command list when creating a command button). It has two parameters:

- **Filter:** The name of the filter to be applied.
- **Skip *n* if not found:** (Optional.) Use this setting to supply a custom message/behavior if the record search finds no matches.

If you leave the 0 value in place, DroidDB applies the default behavior when a search yields no results: that is, it displays the message “Record not found” and quits the macro.

Alternatively, you can suppress the “Record not found” error that normally occurs when no record is found and have your macro skip forward or backward *n* commands in the macro sequence. For example, you could create a custom message box command, and supply a value for *n* that skips to that command when no match is found.

### Option | Filter | Off

Displays all records in the table. It has no parameters.

**Option | Import command****Option | Import**

This command imports the contents of a comma delimited ASCII text file into the current table.

It has the following parameters:

- **No headers:** Select this option if the file contains no headers; that is, if the first line of the file does not contain the table column names.
- **File:** The table column or global variable that stores the pathname of the file to import.

If you do not provide any value for File, DroidDB looks in the current application folder for a text file with the same name as the table associated with the current form. For example, if MyForm uses MyTable, DroidDB will look for MyTable.txt in the current application folder.

**Notes**

- To determine the proper format for an import file, including field order, export the current contents of the table examine the results. (To export a file, open the form on the Android device and select Options-Export.)
- DroidDB considers any of the following date/time formats valid, and imports values that conform to them as is:
  - ◇ YYYY-MM-DD
  - ◇ YYYY-MM-DD HH:MM
  - ◇ YYYY-MM-DD HH:MM:SS
  - ◇ Formats specified via the device's Regional Settings.

### **Option | Recalc command**

This command recalculates the values for all calculated field, lookup, and grid controls in the current record. It works the same as the DroidDB menu option of the same name.

An Option | Recalc command has no parameters.

## Play Sound command

### Play sound *sound*

This command, typically used as a step in a macro sequence, causes the Android device to play a sound. You can select from a set of predefined sounds or you can supply your own .wav file. The .wav file must be in the application folder.

It has one parameter:

- Select from the symbols **OK**, **\***, **!**, **X** and **?** -- each is associated with an event that generates a sound in the standard sound scheme. Or, click [...] and select the desired .wav on the Android device.

In the Macro Builder, click **Play Sound** to hear the sounds on the PC.

## Record commands

These commands manipulate records or the record display in a wide variety of ways. Each command works the same as the DroidDB menu option of the same name. Unless otherwise indicated, they have no parameters.

### Record | Close

Closes the DroidDB application.

### Record | Delete

Deletes the current record.

### Record | Duplicate

Duplicates the current record.

### Record | First

Moves to the first record in the table or filtered list.

### Record | Insert

Creates a new record.

### Record | Last

Moves to the last record in the table or filtered list.

### Record | Next

Moves to the next record in the table or filtered list.

### Record | Previous

Moves to the previous record in the table or filtered list.

### Record | Save

Saves the current record.

### Record | Search *exact matches only, arg, skip n if not found*

Finds the first record (based on the current sort order, and filter if any) whose value in a specified table column matches a search value. It has three parameters:

- **Exact matches only:** Select this option to find exact matches only (case is ignored). For example, if **Fred** were the search value, the result would be the first record in the table that stored the value **Fred** or **fred** in the sort column. **Fred Smith** would not match. If the table does not contain a record with the exact match in the sort column, the result would be the message, "Record not found.", or custom behavior you specify; see Skip if not found below. If you do not select this option, the command finds the first record that begins with the search value, or the next greatest value if no exact matches exist.
- **Arg:** (Optional) The global variable that holds the search value. If you do not select a variable, the form will prompt the user for a search value.
- **Skip *n* if not found:** (Optional) Use this setting to supply a behavior if the record search finds no matches.

If you leave the 0 value in place, DroidDB applies the default behavior when a search yields no results: that is, it displays the message "Record not found." and quits the macro.

Alternatively, you can supply your own message and/or behavior. For example, you could create a custom message box command, and supply a value for *n* that skips to that command when a match is not found.

## Report command

**Report** *report, destination, current record only*

This command launches a report template ( a .drp file).

It has the following parameters:

- **Report:** The report (.drp file) to run.
- **Destination:** You are given the option of sending the report by email or saving it to a file. (The print option is not yet implemented.) There is no “view report” option. To view the report, save it to a file and then use a RUN EXTERNAL command button to view the file.
- **Current record only:** *Available if the application table for the DroidDB form and the top level table for the report layout are the same table.* If this option is selected, the report only contains data from the current record. If this option is not selected (or the application table for the DroidDB form and the top level table for the Report layout are not the same table), the report contains data from all the records in the current table.

### **Return from Macro command**

This command stops the execution of the current macro and returns control to the process that called it. If this statement is included in a submacro, processing returns to the main macro. If included in a main macro, processing returns to the form. It is recommended as the last step in every macro.

## Run External command

### Run External *command, arg, close form*

This command launches an external program and (optionally) passes a value from the current record to the second program. It is not used to launch reports (use the Report command instead) or run other DroidDB forms (use Run Form command).

It has the following parameters:

- **Command:** The program to run.

If this is left blank, then “Arg” is the table column or global variable that contains the name of a document. This could be the file pathname of a .pdf or .wav, for example, or a URL. The Android operating system will find an application to present the document.

If it is not left blank, then it is name of a package to run. Ask the developer of the app what the package name is: for example, the DroidDB package name is `com.syware.droiddb`. The "Arg" is passed to the application as a String called "Parameter" in the application's intent's extra's bundle.

- **Arg:** (Optional) See above.
- **Close form:** If selected, the form closes when the external program is launched.

## Run Form command

**Run Form** *formname, closeform*

This command launches another DroidDB form.

It has two parameters:

- **Formname:** The form to run.
- **Close Form:** Whether or not the current form closes when the second form opens. This option determines whether or not the Back button goes from the second form to the first form, or to a prior location.

## Run Macro command

### Run macro *macro*

This command starts a specified macro as a submacro.

It has one parameter:

- **Macro:** The name of the macro. (The macro must have already been defined in the current form.)

## Select Tab command

### Select tab *num*

This command selects a specified tab on the form.

It has one parameter:

- **Num:** The number that identifies the tab to select. The first tab is identified by 0, the second by 1, the third by 2, and so on. You can supply a number or an expression (page 200) that returns a number.

## Set Focus command

### Set focus *control*

This command puts focus on a form control. You might use it in a validation macro to put focus on a control with failed input so that the user can more easily correct the value, for example.

It has one parameter:

- **Control:** The form control to get focus.

### Tip

This command can trigger a Get Focus or Lose Focus event.

## Show/Hide Control command

**Show/hide control** *control*, Yes (*Show*) or No (*Hide*)

This command, typically used as a step in a macro sequence that is triggered by an event, shows or hides any control on the form. (This application is illustrated in Quick Tour IV).

A Show/hide control has two parameters:

- **Control:** The form control to show or hide.
- **Yes (Show) or no (Hide):** If you select this option, the control is displayed when the command is executed. If you deselect this option, the control is hidden when the command is executed.

## Skip command

### Skip *num*

This command is used to skip forward or back in the macro sequence. Skip commands make it possible to have conditional branches and loops in a macro.

A Skip command has one parameter:

- **Num:** May be a number of steps to skip, or a GO TO clause with a label.
  - ◇ *Number:* The number of commands to skip; positive values skip forward, negative values skip backward. You can supply a number or an expression (page 200) that evaluates to a number.
  - ◇ *GO TO clause:* You can type “Go To” and a label that matches a macro step label elsewhere in the script. When the skip command is executed, processing jumps to the line identified by the label. (See example on page 216.) “Go To” and macro step labels are not case-sensitive.

### Tip

You can double-click the Num field to open a larger area in which to type the expression or Go To clause.

To create a conditional branch, you can use an “if-then-else” expression for *Num* that controls how far to skip depending on values in the current record. For example, the following command would execute the next step in the macro if the value for age in the current record were less than or equal to 0, otherwise it would skip the next step.

```
if age <= 0 then 0 else 1
```

You can even nest one “if” inside another to create “if...else if ... else” expressions, as illustrated below. This sample macro applies a 25% discount if the total is more than \$1000, or a 10% discount if the total is between \$100 and \$1000.

Step	Command	Command parameters
1	Assign	Value: ITEM_1 + ITEM_2 + ITEM_3 Col: TOTAL
2	Skip	Num: If TOTAL > 1000.00 then 0 else if TOTAL >= 100.00 then 2 else 3
3	Assign	Value: TOTAL * 0.75 Col: TOTAL
4	Skip	Value: 1
5	Assign	Value: TOTAL * 0.9 Col: TOTAL
6	Return from macro	

Alternatively, you can create a conditional branch with a GO TO clause; i.e.,

```
if condition then go to label
```

For example, `if age <= 0 then go to ErrorMessage`

Note that there is no 'Else' when you use a GO TO clause; if the condition is not true, no skipping is done and the next step is executed.

Quick Tour IV, “Getting Acquainted with Macros and Events” (page 70) provides another example of conditional branching. The sample script at the beginning of this section (page 216) demonstrates how to create a loop with Skip commands and Go Tos.

## Sleep command

### Sleep secs

This command delays execution of the next step in the macro for a specified number of seconds. In fact, it freezes the entire application for the specified period.

A Sleep command has one parameter:

- **Secs:** The number of seconds to sleep; may be a constant or the result of an expression. You can double-click the field to open a larger area in which to type the expression.

See also **Timer** command.

## SMS Message commands

### SMS Receive Message *Command, Arg*

This command automatically launches a DroidDB form upon receipt of an SMS instant text message that begins with the string `.ddb.` and optionally saves the incoming message to a global variable.

To receive the message, the DroidDB application need not be running.

It has two parameters:

- **Command:** The DroidDB form to launch.
- **Arg:** Global variable in which to store the message. If no argument is supplied here, the form is launched but the message is not saved.

### SMS Send Message *Msg, Phon*

This command sends an SMS instant text message from the DroidDB form to a phone number contained in the current record or global variable.

It has the following parameters:

- **Msg:** The message to be sent; can be a literal string or the result of an expression. As with any expression in DroidDB, you must enclose literal strings in single quotes (') and column names that included spaces in double quotes (").

For example, to create a message template for notifying customers of package shipments, you could specify the following expression (assuming that `firstName` is a table column):

```
'Dear ' & firstName & ', ' & 'Your package was mailed on ' & @now & '.'
```

You can double-click the field to open a larger area in which to type the message.

- **Phon:** The table column or global variable containing the phone number of the intended recipient.

## Sort by command

### Sort by *col*

This command sorts records in the table by values in a table column. You would typically apply this command prior to another macro step that depends on the records being in a certain order.

A Sort by command has one parameter:

- **Col:** The table column to sort by. This column must be indexed (see page 181).

### **Stop Macro command**

This command stops the current macro. If included in a submacro, processing does not return to the main macro. It is typically used when an error condition is detected.

## Timer command

**Timer** *secs, macro*

This command starts another macro after a delay of some number of seconds. Unlike the **Sleep** command which freezes the application, the timer allows the user to work with the form while it counts down. It has two parameters:

- **Secs**: The number of seconds between the time the command is executed and the designated macro is started; may be a constant or the result of an expression. You can double-click the field to open a larger area in which to type the expression.
- **Macro**: The name of the macro to start; must be a macro already defined in the current application.

See also **Sleep** command.



## **Part V: Using DroidDB Applications**

# Using DroidDB Applications

## Overview of Part V

If you used the sample application in Quick Tour, you already know the basics of using DroidDB applications on Android devices. This section contains detailed reference information for using DroidDB applications.

The following topics are discussed:

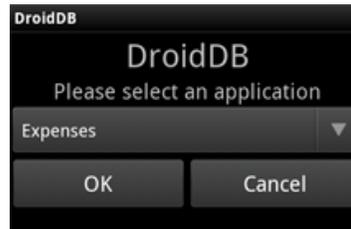
- DroidDB Application Window
  - ◇ Starting DroidDB applications
  - ◇ Menu commands
  - ◇ Exiting DroidDB applications
- Viewing Records
  - ◇ Scrolling through records
  - ◇ Searching for specific records in the table
  - ◇ Using a grid control
- Adding, Editing, and Deleting Records
  - ◇ Creating new records
  - ◇ Editing records
  - ◇ Using scribble boxes
  - ◇ Using image controls to take and store photos
  - ◇ Entering null values
  - ◇ Saving records
  - ◇ Deleting records
- Exporting and Importing Data
  - ◇ Exporting a table
  - ◇ Importing an ASCII text file
  - ◇ Clearing a table
- Remote Synchronization Using mEnable

## Starting DroidDB Applications

DroidDB applications are used on Android devices only. The Android device should be disconnected from the desktop PC.

1. On the Android device, tap  **DroidDB**.

DroidDB asks you to select the application you want to use.



2. Choose the name of the DroidDB application from the drop-down menu. Click **OK**.

DroidDB displays the application form in the application window.



DroidDB applications have two menus – **Record** and **Edit**. You open these menus by tapping the device's Menu button.

For some forms, the keypad is displayed automatically. If you want to dismiss the keypad so that you can see the whole form, tap the Android device's Back button.

## Record Menu Commands

To open the Record menu, tap the device's Menu button, then **Record** .

<b>Command</b>	<b>Function</b>
First	Displays the first* record.
Next	Displays the next* record.
Previous	Displays the previous* record.
Last	Displays the last* record.
Search	Locates a record*.
Save	Saves the current record.
Insert	Creates a new record.
Delete	Deletes the current record.
Close	Closes the application.

\*Based on the current sort/search order.

## Option Menu Commands

To open the Options menu, tap the device's Menu button, then **Options**.

<b>Command</b>	<b>Function</b>
Import	Copies data from an ASCII file to the form's table on your Android device.
Export	Copies the data from the form's table to an ASCII file on your Android device.
Clear	Deletes <b>all records</b> in the form's table. This command <i>does not simply clear the current record</i> ; instead it clears all records in the table.
Recalculate	Recalculates the values in the current record's calculated field, lookup, and grid controls.
About	Displays information in the About box.

## **Exiting DroidDB Applications**

To exit a DroidDB application, touch the device's Menu button > **Record** > **Close** (Close is the last item on the list, you may have to swipe the list of options to bring it into view).

Droid DB saves any changes or new records created since your last save, then closes the DroidDB application window.

## Scrolling Through Records

Once you have created two or more records (or if records have been downloaded from an application on your desktop PC), you can scroll through the records in the table.

The Record menu provides four scrolling commands (First, Previous, Next, and Last).

<b>Command</b>	<b>Function</b>
First	Displays the first table record.
Previous	Displays the previous table record.
Next	Displays the next table record.
Last	Displays the last table record.

## Searching for Specific Records in the Table

DroidDB's Search features scrolls the display to the first record whose value in a column specified by the form designer matches a value you provide. Using the Quick Tour's Expense Tracker application as an example, you could find records by date.

To apply the search feature:

1. Tap the device's Menu button > **Record** > **Search**.

DroidDB displays a dialog box in which you can supply the search value.

2. Enter the search value, then click **OK**.

### Tip

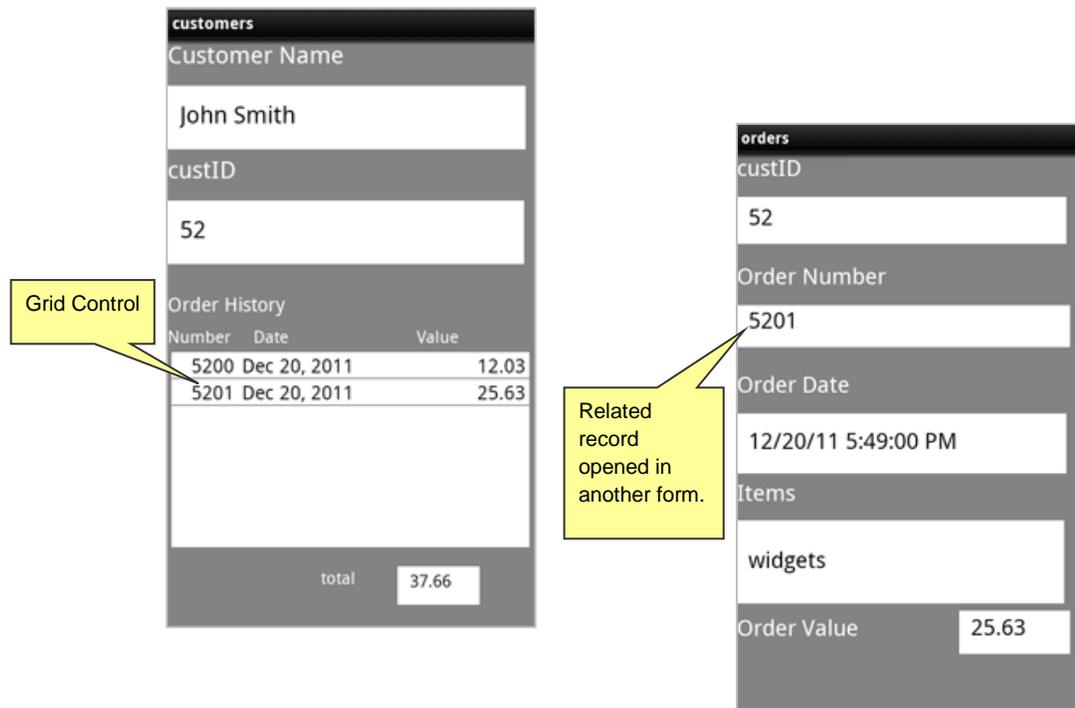
If you are searching on a text column, it is not necessary to enter a complete word. For example, if you entered just **ja**, DroidDB scrolls to the first record whose value in that column begins with those two letters – e.g., Jack, Jane, etc. Text search values are not case sensitive.

DroidDB displays the first record whose value matches your entry. If there are no records with the value you specified, DroidDB looks for the record with the next greater value, or if there are none, it displays the last record in the table.

Note that date/time values are done in reverse order. The newest record is first, and the oldest record is last. Older records are greater than newer records.

## Using a Grid Control

The form designer may have provided a grid control that displays a listing of the records in the current form's table or in a related table. You may be able to select an item in the list to display the full record in the current form or another, as illustrated below. See your form designer for details.



To jump from a grid control item to full record display:

1. In the grid control, long press (press and hold your finger for two seconds) the row for the desired record to select it.
2. Long press the row again to initiate the jump.

Depending on the form design, DroidDB will open another form that displays the selected record.

## **Creating New Records**

To create a new table record, tap the device's Menu button > **Record** > **Insert**.

DroidDB displays a new record (with default values) in which you can make your entries.

## Editing Records

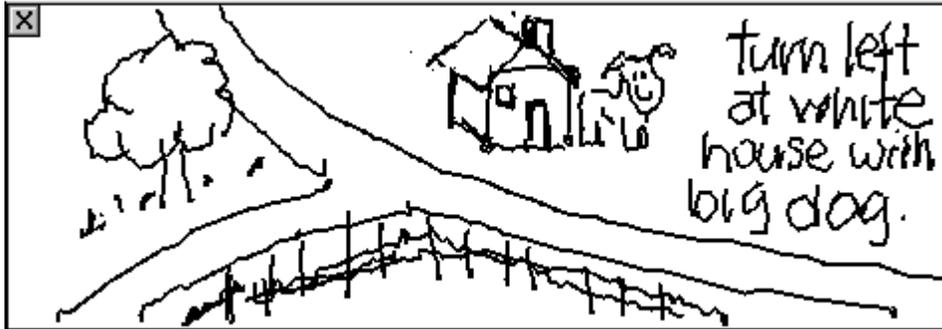
DroidDB allows you to easily edit existing records.

To change an existing field value, simply tap the field and enter or select a new value.

To recalculate the values shown in the current record's calculated field, lookup, and grid controls, tap the device's Menu button > **Options** > **Recalc**.

## Using Scribble Boxes

To make an entry in a Scribble box, simply draw or write inside the box with your finger or the stylus.



To clear the entire scribble and start over, click the X that appears on the left side of the box.

## Using Image Controls to Take and Store Photos

If your Android device has a camera, you can use an image control to take a picture and save it in the current record. All you do is create or open a record, tap the image control, and take the picture using the device's camera application.

To use an Image control to take and save pictures:

1. Tap inside the Image control box. (If you are taking a new picture, tap inside the empty box. If you are replacing an existing picture, click the image to replace it.)
2. The camera application opens. Take the picture as you normally would.

### Tip

To clear a picture from the display, click the X that appears on the left side of the box.

### Note

DroidDB stores the picture according to the form designer's specifications – it either puts the file in the table, or stores the name of the file in the table. In the latter case, DroidDB names the file using the convention, *deviceName\_dateTimeTaken*, and places the file in the application folder (the folder with the same name as the application, at the top level of the Android device's directory structure).

If the picture is stored in the table, it will be an OLE Object (bitmap) in the desktop/server database when the record is synchronized with the desktop PC or server.

## Entering Null Values in a Date/Time or Numeric Field

If you don't want to specify a date in a Date/Time field or a value in a numeric field, you can enter a null value.

1. Make the field the focus by tapping it.
2. If the numeric keypad or date/time picker comes up, tap **Cancel**. (You enter null values directly into the field, not via the calendar dialog or number pad.)
3. Press Delete on the keyboard.

## Saving Records

DroidDB automatically saves new records or changes to records every time you:

- Scroll to another record
- Insert a new record
- Delete a record
- Exit the application

To save your work periodically, you can tap the device's Menu button > **Record** > **Save**.

## Deleting Records

You can delete a single record, or you can delete all records in the table.

### To delete a single record from the table:

1. Display the record in the application window (scroll to it, if necessary).
2. Tap the device's Menu button > **Record** > **Delete**.

DroidDB deletes the record immediately.

### To delete all the records in the table:

- Tap the device's Menu button > **Options** > **Clear**.

DroidDB displays a message asking you to confirm that you want to delete all of the records.

## Exporting a Table to a Text File

On the Android device, you can export the data the form is using to an ASCII comma delimited text file. You can then email your data and/or import it into other applications. This feature is also useful for archiving records.

To export the table, tap the device's Menu button > **Options** > **Export**. DroidDB exports the file.

### Tip

After the table is exported, you may want to delete all records in the table. (Be sure to keep the archived records in a safe place.) If you don't delete the records, and you later import the file back in, you will have two of every record.

### Notes

- DroidDB automatically gives the file the same name as the table (e.g., MyTable.txt) and places it in the application folder on the Android device.
- All date/time values are automatically exported in the following format:

YYYY-MM-DD HH:MM:SS

## Importing an ASCII File to a Table

You can import an ASCII comma delimited text file into a table on your DroidDB device. This file may have been created using the Export command.

1. Make sure that the name of the file to import matches the name of the target table (e.g., MyTable.txt). Also, the import file, target table, and form must all reside in the same DroidDB application folder on the Android device.
2. Start DroidDB on the Android device, and open the application.
3. Navigate to the form built on the target table.
4. Tap the device's Menu button > **Options** > **Import**.

The records are now available to the application.

### Notes

- If a form based on a table is running, such as Myform based on MyTable, DroidDB assumes that the text file to import is named MyTable.txt, and looks for it in the application folder.
- DroidDB considers any of the following date/time formats valid and imports values that conform to them, as is:
  - ◇ YYYY-MM-DD
  - ◇ YYYY-MM-DD HH:MM
  - ◇ YYYY-MM-DD HH:MM:SS
  - ◇ YYYY-MM-DDTHH:MM:SS
  - ◇ Formats specified via the device's Regional Settings.
- To determine the proper format for an import file, including field order, export the current contents of the table and examine the results.

## Clear Table

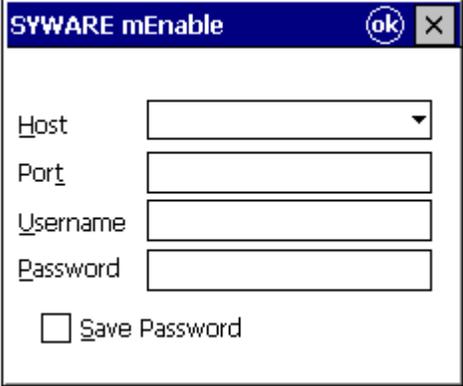
To delete all records in the table, tap the device's menu button > **Options** > **Clear**.

## Using DroidDB with mEnable

SYWARE mEnable makes it possible for you to synchronize data in DroidDB tables on your Android device with tables on a remote database server via a wired or wireless connection including the Internet.

The form designer may have supplied a command button you can tap to make an mEnable connection and initiate synchronization, or they may have incorporated a macro that starts and controls the synchronization “behind the scenes.” Once started, mEnable synchronizes every eligible table in the current DroidDB application on the Android device with tables on the server.

Depending upon how the form designer set up the application, a log on box may appear just the first time you start an mEnable session on the Android device, or it may appear every time, or it may not appear at all.



The image shows a screenshot of a dialog box titled "SYWARE mEnable". The dialog box has a blue title bar with the text "SYWARE mEnable" on the left, an "ok" button in the center, and a close "X" button on the right. Below the title bar, there are four input fields arranged vertically: "Host" (a dropdown menu), "Port" (a text box), "Username" (a text box), and "Password" (a text box). Below these fields is a checkbox labeled "Save Password".

If it does appear, complete the logon box as follows:

- **Host:** The IP address of the server (for example, 208.77.50.186).
- **Port:** Leave blank (DroidDB provides this value automatically).
- **Username and Password:** Enter the values required for your database connection.
- Click **OK**.

## **Part VI: Synchronizing Tables**

# Table Synchronization (Business Edition Only)

## Overview of Part VI

When you synchronize tables, changes made to one are updated in the other. This section explains how to set up and take advantage of this feature.

Topics covered in this section:

- Creating a Synchronization Configuration
- Using Timestamp-based Synchronization
- Synchronizing Multiple Handhelds
- Synchronizing Tables On Command Using DroidDB
- Synchronizing Tables Using DDB\_SYNC
- Synchronizing Tables Remotely Using mEnable
- Synchronizing Tables with Excel Spreadsheets

## Creating a Synchronization Configuration for DroidDB Tables and ODBC-Enabled Databases (Business Edition Only)

Synchronization enables you to share data between DroidDB applications on Android devices and ODBC-enabled database tables on desktop PCs or remote servers.

Any table created using DroidDB's **File > Create Table** or **File > Download Table** options can be synchronized with any ODBC-enabled table on the desktop PC or server.

You can set up synchronization between one desktop/server table and a corresponding table used on any number of Android devices. Each Android device must be uniquely named.

When you create a table using **File > Download Table**, you typically specify the synchronization options during that process. You can use the instructions in this topic to modify those settings, or (less commonly) to create synchronization settings for tables created using **File > Create Table**.

Once you have configured the synchronization options using the instructions in this topic, you can use the DroidDB development environment or `DDB_SYNC.exe` to synchronize tables on the desktop PC with those on a connected Android device, or your DroidDB application users can synchronize their local tables with a remote servers using `mEnable`. `mEnable` is a separate product available from SYWARE.

### Prerequisites

Before you can set up and apply synchronization:

- The first column of the desktop PC table must be an "OID" column. (DroidDB automatically adds a corresponding column to the table on the Android device and uses the columns to keep records in the two tables in sync.) If you have not already done so, open the desktop table in the application that was used to create it, e.g., Microsoft Access. Add a new first column to the table with the following specifications (use default values for any other settings):
  - ◇ Field name: **OID**
  - ◇ Data Type: **Number**
  - ◇ Field Size: **Long Integer (for Access tables)**
  - ◇ Indexed: **No**

When creating new records in the desktop PC table, always set the OID column to null (blank) or zero.

- You must have a Level 2 read/write ODBC Driver for the desktop PC database and an ODBC Data Source built on that ODBC Driver. See the documentation for your ODBC Driver (provided by your database vendor) for installation and setup instructions.
- **If any of the Android devices to be synchronized are "portable devices"**: When connected to the desktop PC, an Android device may appear to Windows as a storage card (with a drive letter) or as a portable device (without a drive letter). The synchronization process for "portable devices" uses the "lite" version of `mEnable` server. That means that you, the form designer, must incorporate the "`mEnable Synchronize`" command (page 241) into the form – either as a command button that the user touches to initiate the synchronization, or as a step in a macro that initiates synchronization automatically. Note: Unlike the full version of `mEnable` for wireless synchronization, which is purchased separately, the 'lite' version for connected portable devices is provided as part of the DroidDB Business Edition.

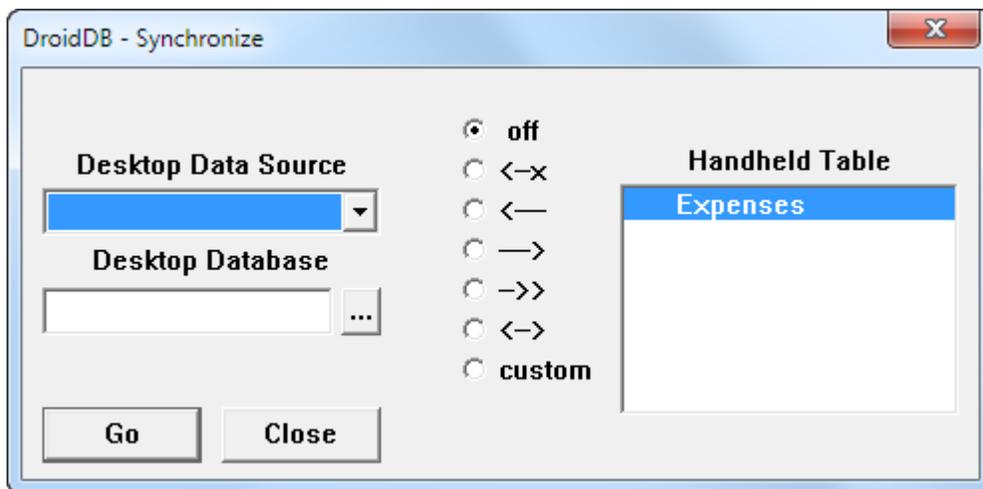
**Note**

The Android device you are using to develop the application must be connected to the desktop PC throughout the following procedure.

To specify synchronization options:

1. Open the DroidDB development environment on your desktop PC.
2. Pick the DroidDB application that contains the table(s) you want to synchronize.
3. If you plan to synchronize the desktop PC table with multiple Android devices, each Android device – including the development device – must be uniquely named. If you have not already done so, select **File > Handheld Name** and supply a name. You need to do this just once for each Android device.
4. Select **File > Synchronize**.

The DroidDB Synchronize dialog box opens.



5. In the **Desktop Data Source** listbox, select the ODBC data source that reads the database on the desktop PC.
6. In the **Desktop Database** field, select the desktop database that contains the tables you want to synchronize.

**Note**

If the data source that you selected in Step 4 intrinsically identifies this database, leave the Desktop Database field blank—see your database documentation.

**Tip**

It is assumed that most of the tables on the handheld will be synchronized with tables contained in a single desktop database (the one you pick here). However, you can override this selection for individual handheld tables using the Custom Synchronization options described in the following topic (on page 292).

7. In the **Handheld Table** listbox, select the table on the Android device to synchronize.
8. Click a radio button to specify how DroidDB will synchronize data between the two tables:

Use this radio button	To do this...
off	Prevent synchronization of the handheld table with the desktop PC table. Any changes made to records in one table are NOT applied the other. This is the default.
←x	Copy new or modified records from the handheld table to the desktop PC table, and delete them from the handheld table (e.g., move records from the handheld to the desktop PC).
←	Apply all changes to records in the handheld table to the desktop PC table. Changes in the desktop PC table are NOT applied to the handheld table.
→	Apply all changes to records in the desktop PC table to the handheld table. Changes in the handheld table are NOT applied to the desktop PC table.
⇒	Publish. Replace all records in the handheld table with all records from the desktop PC table.
↔	Apply all changes to records in the handheld table to the desktop PC table and vice-versa (e.g., all changes appear in both tables).
custom	<p>Opens the [table] Synchronization dialog box, which enables you to further fine-tune the synchronization options. Refer to the following topic, "Custom Synchronization Options" on page 292 for details.</p> <p>Tip: Even if you don't modify the settings, this dialog box is useful for viewing the specific actions associated with each of the standard options listed above (off, ←, etc.)</p>

Notice that DroidDB puts the synchronization symbol next to the table name. (A straight bar, —, indicates that you have specified a custom option.) Whenever synchronization occurs, DroidDB applies the option indicated by that symbol.

- Repeat Steps 7 and 8 for every handheld table that you want to synchronize with a table on the desktop PC or server.
- Click **Close** to save your settings and close the dialog box; or click **Go** to save your settings and run synchronization now.

If you run synchronization now, the next steps depend upon whether your Android device appears to the desktop PC as a storage card (e.g., has a drive letter associated with it), or as a portable device (e.g., does not have a drive letter associated with it). See the topic "Synchronizing Tables On Command Using DroidDB (Business Edition Only)" on page 296 for details.

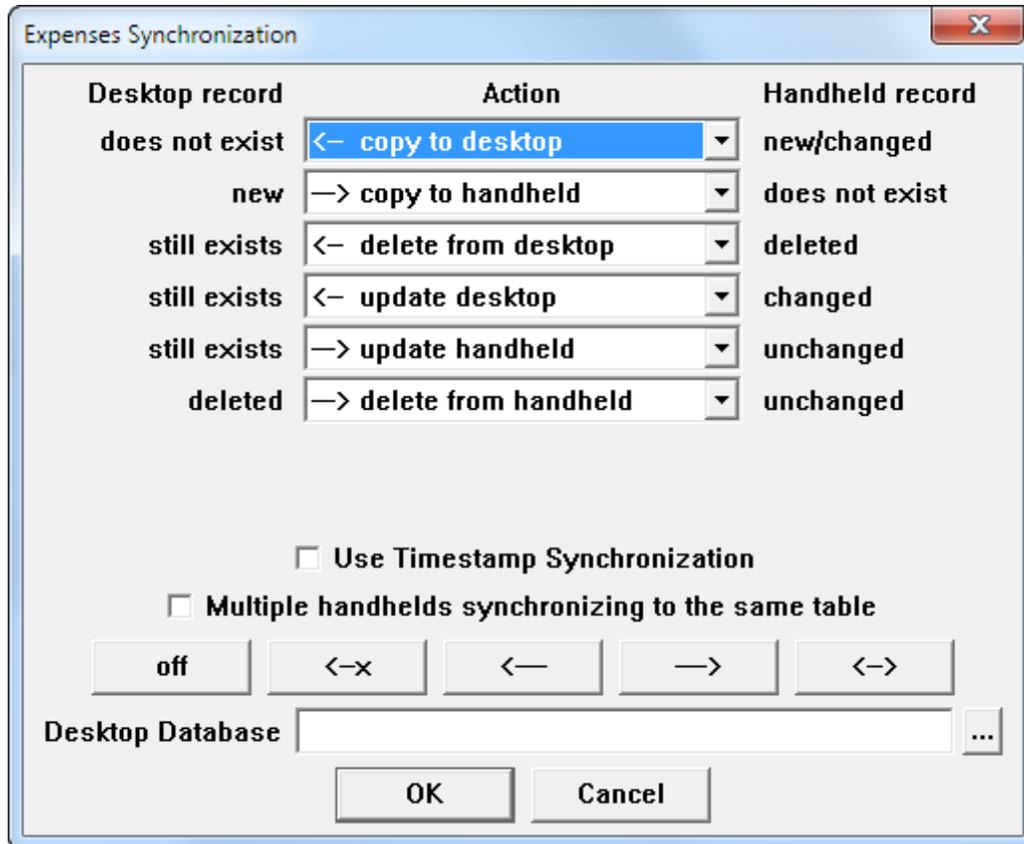
From now on (or until you change the settings), DroidDB synchronizes the tables according to the settings you just specified.

#### Note

When creating new records on the desktop PC, always set the value in the OID column to null (blank) or zero.

## Custom Synchronization Options (Business Edition Only)

If you select the “custom” option in the Synchronize dialog box (see previous topic), DroidDB displays a second dialog box that enables you to fine-tune how updates are copied from one table to another.



You can customize the settings by selecting actions for specific conditions (e.g., a new or changed record on the handheld does not exist on the desktop) using the drop-down arrows at the right side of the Action column.

If you change your mind and want overwrite your customized settings with any of the standard “factory” settings, click the corresponding button near the bottom of the dialog box (off, <-x, <-, -> or <->). The standard synchronization actions associated with each of these buttons are described near the end of the previous topic, “Selecting Table Synchronization Options.”

### Timestamp Synchronization

If you want to use timestamp-based synchronization, select the Use `TIMESTAMP` Synchronization checkbox. Timestamp-based synchronization is faster and provides greater control than regular synchronization but requires additional effort to maintain the timestamps. See the following topic, “Using Timestamp-Based Synchronization,” for details.

### Multiple Handhelds Synchronizing to the Same Table

You can synchronize one desktop PC or server table with any number of Android devices. The synchronization settings you specify on the development handheld are automatically stored with the DroidDB application, and thus included with any applications you distribute to other users.

All you need to do is select the “Multiple handhelds synchronizing to the same table” option here, and give each of the Android devices a unique name. To name a device, you must attach it to the desktop PC and name it using DroidDB’s File > Handheld Name command.

**Notes**

- If the “Multiple handhelds synchronizing to the same table” option is grayed out, the currently attached device does not have a name .
- If you specified multi-handheld synchronization when you created the handheld table using File > Create Table, DroidDB turns on this option for you automatically.

**Desktop Database**

Use this field if the handheld table synchronizes with a table in a database other than the one selected in the main Synchronize dialog box.

## Using Timestamp-Based Synchronization (Business Edition Only)

During normal synchronization, DroidDB compares every record in a desktop PC table to every record in the handheld table to determine what has changed. Even if only a few records have been added or changed, this process can take a long time because the synchronizer must check all records.

If you know that only a few changes are likely between synchronizations, you can minimize synchronization time using a technique called Timestamp-Based Synchronization.

This technique depends on you, the designer, implementing a mechanism to indicate which records on the desktop PC are new or changed. You do this by adding a date/time column named `TIMESTAMP` at the end of the desktop PC table. Additionally, whenever a record in the desktop PC table is added or modified, you must set the `TIMESTAMP` value for that record to the current date and time.

Do NOT make any modifications to the table or records on the handheld to accommodate Timestamp-Based Synchronization. DroidDB knows what records it has changed there. In particular, do NOT add a Timestamp column to the table on the handheld.

Setting the `TIMESTAMP` column on the desktop PC for new or modified records is all you need to do; during synchronization, DroidDB reads these timestamps and resets them as necessary.

Using timestamps is optional and involves certain tradeoffs:

- If you use timestamps, note the following:
  - ◇ You must set the timestamps on updated desktop PC records. Failure to set the timestamps causes DroidDB to ignore desktop PC updates.
  - ◇ You must set the timestamps on inserted desktop PC records. Failure to set the timestamps causes DroidDB to delete these records from the desktop PC table at synchronization time.
- If you do not use timestamps, note the following:
  - ◇ Synchronization is slower.
  - ◇ If a record is updated on both the handheld and the desktop PC, the handheld record overwrites the desktop PC record. (If you use a timestamp column, DroidDB asks which record to use.)
  - ◇ If a record is updated on the desktop PC and deleted from the handheld, DroidDB deletes the record from the desktop PC. (If you use a Timestamp column, DroidDB copies the record from the desktop PC to the handheld.)

### About timestamp synchronization and existing desktop PC tables:

If you enable timestamp synchronization for a desktop PC table that already exists, you must manually add a Timestamp column to the table before synchronizing it. The column must meet the following requirements:

- It must be the last column in the table
- It must be named `TIMESTAMP`
- Its data type must be defined as Date/Time

If you disable timestamp synchronization for a desktop PC table, you must manually remove the Timestamp column from the table before synchronizing it again.

## Synchronizing Multiple Handhelds (Business Edition Only)

You can synchronize a table on the desktop PC or server with the corresponding table used on any number of Android devices.

To set up synchronization for multiple Android devices:

1. Set up synchronization between the desktop PC table and the DroidDB table on your development Android device, either during the download process (see page 101) or following the instructions in the topic, "Creating a Synchronization Configuration for DroidDB Tables and ODBC-Enabled Databases (Business Edition Only)," on page 289. In either case, be sure to name the Android device as described.

If using the first method, be sure to respond "Yes" when DroidDB asks if you want to set up synchronization between tables and if you will be synchronizing to more than one handheld.

If using the second method, be sure that the **Multiple handhelds synchronizing to the same table** option is checked in the Custom Synchronization dialog box (see page 292).

2. When you save your DroidDB application using the development environment on the desktop PC, the table, form, and synchronization settings are all saved in the application's folder on the attached Android device. To install the application on other devices, create and install the distribution files as explained on page 196. Or, simply copy the application's folder from the top-level directory of Android device you are using to develop the application to the top-level directory of another Android device.
3. Assign a unique name to each Android device that will be synchronizing to the desktop/server table:
  - Attach the Android device to the desktop PC.
  - Start the DroidDB development environment on the desktop PC and open the DroidDB application.
  - Select **File > Handheld Name** and enter a unique name for the device.
  - Close DroidDB.

### Note

You need do this just once for each device, not every application.

You can now synchronize the desktop/server table with the tables on the Android devices using the instructions on the following pages.

## Synchronizing Tables On Command Using DroidDB (Business Edition Only)

Once you've set up the synchronization configuration for a table on the desktop PC or server and a table on the Android device, keeping the two sets of information up-to-date with one another is a simple process.

### Note

Two additional methods are available for performing synchronization:

- Working on the desktop PC, you can use `DDB_SYNC.EXE` to synchronize all eligible tables on a connected Android device with the desktop PC (see page 297).
- Working on the Android device, you can use the "full" version of mEnable to synchronize all eligible local tables in the current DroidDB application with the server database over a wired or wireless connection including the Internet (see page 298) .

To synchronize tables on command using the DroidDB development environment:

1. Connect the Android device to the desktop PC.
2. Open DroidDB on the desktop PC.
3. Pick the DroidDB application that contains the table you want to synchronize.
4. Select **File > Synchronize**.
5. Click **Go**.

The remaining steps depend upon whether your device is recognized by the desktop PC as a storage card (has a drive letter associated with it), or as a portable device (no drive letter associated with it).

**Storage card:** As DroidDB synchronizes all of the tables in the current application, it displays progress messages on the desktop PC and notifies you if there is a problem.

**Portable device:** DroidDB starts a special version of the mEnable software on the desktop PC and notifies you that it is waiting for you to start synchronization on the Android device. Go to the Android device and start the DroidDB application that has the tables to be synchronized. Click the button or start the macro that the form designer has provided to initiate mEnable synchronization. As mEnable synchronizes all of the tables in the selected application, it displays progress messages on the Android device and lets you know if there is a problem. mEnable stops automatically when the process is done.

### Note

When creating new records in the desktop PC table, always set the OID column to null (blank) or zero.

## Synchronizing Tables Using DDB\_SYNC (Business Edition Only)

DDB\_SYNC.EXE makes it possible to synchronize tables without opening the DroidDB development environment.

The process differs depending upon whether your Android device is recognized by the desktop PC as a storage card (has a drive letter associated with it), or as a portable device (no drive letter associated with it). If the connected Android device is recognized as a storage card, DDB\_SYNC synchronizes *all* tables in *all* DroidDB applications on the device. If the device is recognized as a portable device, DDB\_SYNC synchronizes *all* tables in *just the selected DroidDB application*.

This program is included as part of the distribution files for use by your end-users. Refer to the topic, "Creating Distribution Files," on page 196 for details.

### Notes

- Before running DDB\_SYNC.EXE, you must have already set up the synchronization options using the DroidDB development environment, as described in the topic, "Creating a Synchronization Configuration for DroidDB Tables and ODBC-Enabled Databases," on page 289.
- You must be working on the desktop PC with the Android device connected.

To run DDB\_SYNC.EXE:

- On the desktop PC, click the Windows **Start** button, then **All Programs > SYWARE DroidDB > DroidDB Synchronize**.

The remaining steps depend upon whether the connected Android device is recognized by the desktop PC as a storage card or as a portable device:

**Storage card:** As DDB\_SYNC synchronizes all of the tables on the device, it notes each table with a message displayed on the desktop PC. If it encounters any problems (e.g., a table is missing), it stops the process and displays a dialog box on the desktop PC for your response.

**Portable device:** DroidDB starts a special version of the mEnable software on the desktop PC and notifies you that it is waiting for you to start synchronization on the Android device. Go to the Android device and start the DroidDB application that has the tables to be synchronized. Click the button or start the macro that the form designer has provided to initiate mEnable synchronization. As mEnable synchronizes the tables in the application, it displays progress messages on the Android device and lets you know if there is a problem. mEnable stops automatically when the process is done.

## **Synchronizing Tables Remotely Using mEnable (Business Edition Only)**

With mEnable, a separate SYWARE product, DroidDB users can synchronize tables on their Android devices with tables on a server over a wired or wireless connection, including the Internet.

The mEnable software must be installed and running on the server. No additional software is needed on the Android device, since mEnable is included in the DroidDB runtime software.

When you build the DroidDB application, you must configure the synchronization options as described in the instructions beginning on page 289 . You must also incorporate the “mEnable Synchronize” command (page 241) into the form – either as a command button that the user touches to initiate the synchronization, or as a step in a macro that initiates synchronization automatically.

When the user or the macro initiates the synchronization, mEnable establishes a connection with the server and executes full synchronization between all eligible local tables in the current application and the server database.

## Synchronizing Tables on the Android Device with Excel Spreadsheets on the Desktop PC (Business Edition Only)

It is possible, although not recommended, to use Excel files as the desktop database and synchronize them with tables on the DroidDB device using DroidDB synchronization.

This requires some set up:

- DroidDB does not work with Excel files by default, but you can change this by editing the .INI file that is located in the application folder. (You may wish to make a backup copy of this file before changing it.) Using a text editor such as Notepad, open the file and locate the line that reads [VICESYNC]. Just below that line, insert a new line that reads: Excel=1
- When creating an Excel ODBC Data source, but sure to click on “Options” and turn off the “Read Only” option. (“Read Only” is on by default.)
- You must define the table in the spreadsheet. To do this:
  - ◇ Open the spreadsheet in Excel.
  - ◇ Select the region containing the data.
  - ◇ Select **Insert > Name > Define** from the menu bar.  
(For more details, consult your Microsoft Excel documentation.)

When using Excel as your desktop PC database, keep in mind the following limitations:

- If you create a new record in the Excel file, it must not duplicate a record that already exists in the table in the Excel file.
- Record deletions on the handheld cannot be propagated to the desktop.