

# **FirstClass 7 Rapid Application Developer 2 Guide**

Centrinity Inc.

905-762-6000 or

Sales: 1-800-763-8272

Web: [www.centrinity.com](http://www.centrinity.com)

Email: [support@centrinity.com](mailto:support@centrinity.com)

Printed in Canada

SOF3900.0A

Copyright © 1997, 2002 by Centrinity Inc.

## **Notices**

You must accept the FirstClass License Agreement before you can use this product. If you do not accept the terms of the License Agreement, do not install the software and return the entire package within 30 days to the place from which you obtained it for a full refund. No refunds will be given for returned products that have missing components.

Information in this document is subject to change without notice. Certain features and products described in this document may not be currently available in all geographic regions. Distribution or reproduction of this document in whole or in part must be in accordance with the terms of the License Agreement.

All rights reserved. FirstClass is a registered trademark of a Centrinity Inc. subsidiary used under license. Centrinity, the Centrinity logo, and the FirstClass logo are trademarks of Centrinity Inc. All other trademarks are property of their respective owners.

This edition applies to release 7.0 of FirstClass and to all subsequent releases and modifications until otherwise indicated in new editions. This document is bound by international copyright law and the FirstClass Software License Agreement and Limited Warranty included with every FirstClass product.

## **Technical support**

Telephone technical support is available to registered administrators at the following numbers:

Toll free in North America: 1-800-346-9180

Toronto: 905-415-7060

International: +353-61-725-200

Online support questions may be directed to [support@centrinity.com](mailto:support@centrinity.com).

# Contents

## Figures vii

## Introduction

### 1 Overview 3

- Uses for FirstClass RAD applications 4
- About this book 5
  - Who should read this book 6
  - What you should already know 6
  - Documentation conventions 6
- Coding conventions 7
- ODBC drivers 7
  - ODBC error messages 8
  - FirstClass RAD error messages 8
- New in this release 8

### 2 Administrator tasks 9

- Planning your FirstClass RAD applications 9
- Creating your FirstClass RAD applications 10
  - Preparing your FirstClass settings document and resources 10
  - Flowcharts of FirstClass RAD applications 10
  - General steps for creating a FirstClass RAD application 12
- Maintaining your FirstClass RAD applications 13

### 3 FirstClass RAD concepts 15

- Managing projects 15
- Using forms and resources 18
- Working with databases 19
- How FirstClass RAD forms and databases work together 19
- Using source code 21
- Distributing completed applications 22
- Branding your application 22
- Running FirstClass server applications 23

### 4 Coding concepts 25

- FirstClass specific information 25
  - Field arrays 26
  - Box arrays 27
  - Fixed and expanding lists 28
  - Statistics events 28
- BASIC programming overview 30
  - Object Oriented programming 31

	Event procedures	31
	Coding conventions	31
	Combining multiple lines of code	31
	Continuing code on a new line	31
	BASIC programming and FirstClass RAD	32
	Form modules	32
	Code modules	32
	Database application basics	33
	Connections	34
	Statements	34
	Columns	34
<b>5</b>	<b>Installing FirstClass RAD and logging in</b>	<b>37</b>
	Windows	37
	System requirements	37
	Installing FirstClass RAD	37
	Mac® OS	38
	Logging in to FirstClass RAD	38

## **Planning and creating FirstClass RAD applications**

<b>6</b>	<b>Planning your FirstClass RAD applications</b>	<b>41</b>
	Planning questions	41
<b>7</b>	<b>Creating your FirstClass RAD applications</b>	<b>43</b>
	Creating your FirstClass settings file and resources	43
	Creating your settings document	43
	Creating your forms	44
	Designing your forms	44
	Adding a new form to your settings document	45
	Drawing fields on your form	46
	Setting field attributes	48
	Working in the Project Manager	49
	Creating a nondatabase application	49
	Creating a bound-column database application	50
	Creating a nonbound- column database application	52
	Recording application information	54

## **Maintaining FirstClass RAD**

<b>8</b>	<b>Monitoring FirstClass RAD applications</b>	<b>57</b>
	Loading and unloading an application	58

## **Tutorials**

<b>9</b>	<b>Tutorial 1: Disk Usage utility (nondatabase)</b>	<b>63</b>
	Tutorial materials	63
	Assembling the project	64
	Adding a new form to your settings document	64
	Adding fields to your form	65
	Setting field attributes	65
	Assigning attributes	66
	Creating a new project	66
	Working with forms and fields	67
	Adding a form to your project	67
	Naming forms and fields	68
	Programming your project	70
	Showing the form with Sub Main()	70
	Running the Disk Usage utility in FirstClass RAD	71
	Retrieving disk use information	72
	Forming compound strings	73
	Creating a subroutine	74
	Calling your subroutine	76
	Formatting output	78
	Programming the Refresh button	80
	Running the application	81
	Programming the Exit button	81
	Formatting output (continued)	82
	Building and installing your application	84
	Building your project	84
	Installing additional application icons	85
	Extra practice	87
<b>10</b>	<b>Tutorial 2: User Absence form with bound columns (database)</b>	<b>89</b>
	Tutorial materials	90
	Assembling the project	90
	Adding a new form to your settings document	90
	Adding fields to your form	91
	Setting field attributes	91
	Assigning attributes	92
	Setting the Absence type list	92
	Creating a new project	93
	Working with forms and fields	93
	Adding a new form to your project	93
	Naming forms and fields	94
	Creating a Microsoft Access database	97
	Building your database table	97
	Configuring ODBC	98
	Using the ODBC Driver Manager	98
	Connecting a project to a database	100

	Adding the Requests Table to your project	101
	Binding fields to columns	102
	Adding your table to the data environment	102
	Binding form fields	104
	Programming your project	105
	Showing the form with Sub Main()	105
	Creating a new record	107
	Inserting a new temporary record	107
	Confirming an inserted record with the Update command	108
	Programming the Submit button	109
	Programming the Cancel button	110
	Assigning the 'txtUserName' field	111
	Building and installing your application	112
	Building your project	112
	Extra practice	113
<b>11</b>	<b>Tutorial 3: Administrator Absence form with non-bound columns (database)</b>	<b>115</b>
	Tutorial materials	116
	Assembling the project	116
	Adding a new form to your settings document	116
	Adding fields to your form	117
	Setting field attributes	118
	Assigning attributes	118
	Adding fields to the Absence Detail form	120
	Assigning field attributes	120
	Setting the Absence Type list	120
	Setting the Absence Approval list	121
	Creating a new project	121
	Adding the Absence List form to your project	122
	Naming forms and fields	123
	Connecting your project to a database	125
	Creating a connection to the Absence database	125
	Adding the Requests table to your project	126
	Programming your project	127
	Running your application	130
	Displaying icons in your list	131
	Assembling Project II – adding a second form	132
	Adding the Absence Detail form to your project	132
	Naming forms and fields	133
	Binding fields to columns	134
	Adding your table to the data environment	135
	Binding fields	135
	Programming Project II – integrating your forms	137
	Programming buttons on the List form	137

Programming the Exit button	138
Programming the buttons on the Absence Detail form	138
Running your application	139
Building and installing your application	140
Building your project	140
Extra practice	141

## **Appendix**

### **A FirstClass Designer field attributes 145**

### **B FirstClass RAD error messages 155**

Core error messages (1 – 107) 155

Database error messages (500 – 509) 166

FirstClass error messages (1000 – 1018) 168

## **Index 171**





# Figures

Connection tab	18
Application Monitor	58
Database object hierarchy	33
Designer field attributes	48
Designer field categories	47
Designer form - Database Hitlist	45
Field array for 100 checkboxes	26
FirstClass Application Monitor	54
FirstClass RAD database application flowchart	12
FirstClass RAD environment	4
FirstClass RAD nondatabase application flowchart	11
Form properties	46
Forms tab	17
Module tab	17
Project tab	16
Table of Buttons fields, Extensions, and Tab Controls fields	149
Table of Graphics fields	145
Table of Lists fields and Selectors fields	151
Table of Text & Numbers fields	147
Table tab	18



# Introduction



# Overview

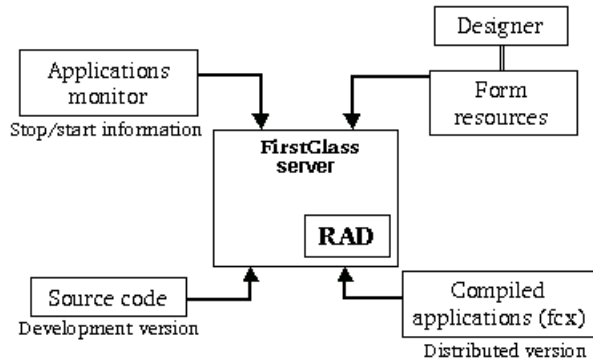
FirstClass® Rapid Application Developer 2 (FirstClass RAD) is a full-featured rapid application development tool for building intranet applications and powerful database interfaces, letting you enhance client functionality. FirstClass RAD resides on your FirstClass server, so it is part of a single, comprehensive integrated system based on the FirstClass core architecture. This makes FirstClass RAD both simple and powerful and allows you to create a variety of applications, which can then be distributed to a large user base without having to install them on client machines.

FirstClass RAD extends the functionality of the FirstClass server by providing the ability to run applications on the server central processing unit (CPU). These applications can then be accessed and developed using the FirstClass client. Unlike other database solutions, the end user needs nothing more than the FirstClass client to take advantage of the database. This method gives the application access to the power, resources, and databases available on the core server without installation or configuration on the client machine.

Because of the cross-platform nature of the FirstClass client, developing an application once makes it available on all supported FirstClass platforms simultaneously. Further, since you have access to the FirstClass environment, you can also tie into a database in order to create users on FirstClass or change permissions or access levels inside the FirstClass system.

Let's take a look at a diagram of how different components are managed in FirstClass RAD:

## How Applications Are Managed



To create FirstClass RAD applications:

- create the required forms using FirstClass Designer
- use FirstClass RAD to program the event procedures for the objects on the forms.

Event procedures run when an event occurs (for example, users click a button or fill in a field on a form).

Although FirstClass RAD must be installed on a Windows® NT server, you just need a FirstClass Windows or Mac™ OS client to run it.

### Uses for FirstClass RAD applications

There are many ways you can use FirstClass RAD applications to enhance your FirstClass client functionality. Some examples are:

- letting users see their disk usage
- linking to customer/contact/sales databases
- integrating user information with external databases
- extracting database information and incorporating it into automatically generated messages
- importing/exporting FirstClass information to other applications
- providing quick user/change password ability

- gathering and processing system and web statistics on the fly
- automating repetitive tasks
- creating/parsing batch administration scripts
- starting and stopping FirstClass Internet Services remotely.

## About this book

This book provides an overview of FirstClass RAD and how you can use it to:

- build intranet applications
- build powerful database interfaces

This book also provides:

- a general overview of the steps involved in creating a FirstClass RAD application

**Note** For information on how to create an application see Chapter 7, "Creating your FirstClass RAD applications" and "Creating FirstClass RAD applications" in our online help.

- BASIC language concepts, database concepts, and coding procedures used within FirstClass RAD

**Note** This book should not be considered a programming manual. For information on programming, we recommend you read additional materials.

- tutorials that allow you to work through different application scenarios.

The Language reference describes the commands, methods, attributes, and so on, that you can include in your BASIC code for FirstClass RAD applications. You can find the FirstClass RAD language reference, in the FirstClass RAD language application on your FirstClass RAD Desktop, or our online help.

For sample applications, please refer to your FirstClass RAD Project Manager. The Systeminfo example project, also inside

the FirstClass RAD Project Manager, has fully commented code examples.

This book is partitioned into the following five sections:

- Introduction

This section provides conceptual information about FirstClass RAD and the administrator's role. This section also provides requirements and instructions for installing FirstClass RAD components.

- Planning and creating FirstClass RAD applications

This section provides questions you should answer before you create your FirstClass RAD applications and information on how to use FirstClass RAD setting files, forms, and Project Manager.

- Maintaining your FirstClass RAD environment

This section provides an understanding of the FirstClass RAD Application Monitor and how to load and unload applications from the FirstClass server.

**Who should read this book**

This book is intended for novice programmers, professional developers, and FirstClass administrators who need to add customized functionality to their FirstClass environment.

**What you should already know**

We assume you are familiar with:

- the FirstClass server and FirstClass Designer
- your operating system
- the fundamentals of BASIC coding, Windows® SQL, database design, and creating ODBC data sources (open database connectivity) using Windows NT®.

**Documentation conventions**

We use certain documentation conventions for menu items and variables in this guide.

*Menu items*

Each level of menu items is separated by >. For example, the Clear item under the Edit menu is shown as Edit > Clear.



*Variables*

Text in *italics* indicates arguments, variables, or other information for which you must type your own value.

## Coding conventions

FirstClass RAD uses standard coding conventions in its syntax descriptions, notes, and code examples:

<b>Description</b>	<b>Examples</b>
Words in bold typeface indicate language reserved keywords.	<b>For, Int, Dim, Print</b>
Items that appear inside square brackets are optional.	[, <i>msgboxtype</i> ], [ <i>stringexpression</i> ]
Curly braces and a bar (pipe) indicate mandatory choices between a number of items.	{Goto <i>label</i>   Resume Next   Goto 0}
Code examples always appear in the Zurich BT font.	Sub Click() StringField1002.SetFocus End Sub

1

## ODBC drivers

FirstClass RAD uses standard Open DataBase Connectivity (ODBC) drivers installed on your Windows computer. Drivers include:

- Microsoft® Access
- DBase™
- Microsoft® Excel
- Microsoft® FoxPro
- Microsoft® SQL
- FileMaker® Pro (5.5 or higher)
- Oracle®
- Borland® Paradox
- Sybase® Inc.

## New in this release

- text.

Since these drivers can be installed by a variety of operating systems, they may vary on each machine. Platforms on which you can install FirstClass RAD include Windows 95, Windows 98, Windows NT, Windows® XP.

### **ODBC error messages**

FirstClass RAD uses standard ODBC error messages. Definitions of these error messages can be found in any Microsoft documentation dealing with ODBC, or on ODBC web sites on the Internet. These error messages will vary depending on the manufacturer of your ODBC driver.

### **FirstClass RAD error messages**

For a complete list and description of FirstClass RAD error messages, see Appendix B, "FirstClass RAD error messages" on page 155.

## **New in this release**

- character sets are now automatic based on the System Character Set of the server.

# Administrator tasks

As the FirstClass administrator, you are responsible for creating and distributing FirstClass RAD applications. Of course, if you have a large system to maintain, you may want to delegate any FirstClass RAD work to a subadministrator or knowledgeable user. Keep in mind, if you do this, you must make sure that your FirstClass RAD developer has the appropriate administrator permissions and privileges to complete this work. For example, he must have subadministrator privileges on the active server.

In this chapter, we provide a high level overview of the three main tasks for which the FirstClass RAD administrator is responsible: planning your FirstClass RAD applications, creating your FirstClass RAD applications, and maintaining your FirstClass RAD environment.

**2**

## Planning your FirstClass RAD applications

Planning is one of the most important stages when creating a FirstClass RAD application. There are several questions you must answer before beginning development. Among other, these questions should be addressed:

- the type of application you require for a particular task
- the inclusion of any databases
- what types of forms and other resources are required.

In Chapter 6, "Planning your FirstClass RAD applications", we discuss these issues in greater detail.

## Creating your FirstClass RAD applications

After planning your application, the next step is creating your application. In this section, we present an overview of the different tasks required to develop your FirstClass RAD application. Details on how to perform these different tasks are presented in Chapter 7, "Creating your FirstClass RAD applications".

### Preparing your FirstClass settings document and resources

All aspects of the project are configured from within FirstClass RAD, with the notable exception of editing the form resources contained in the FirstClass settings document. Forms are drawn and assembled in FirstClass Designer. For information on FirstClass Designer and how to create forms, see *FirstClass Designer*. Once the forms have been created they may be integrated into the FirstClass RAD project by logging into the FirstClass server with a settings document, which contains the correct forms.

There are four main tasks that you must perform when preparing your FirstClass settings document and resources:

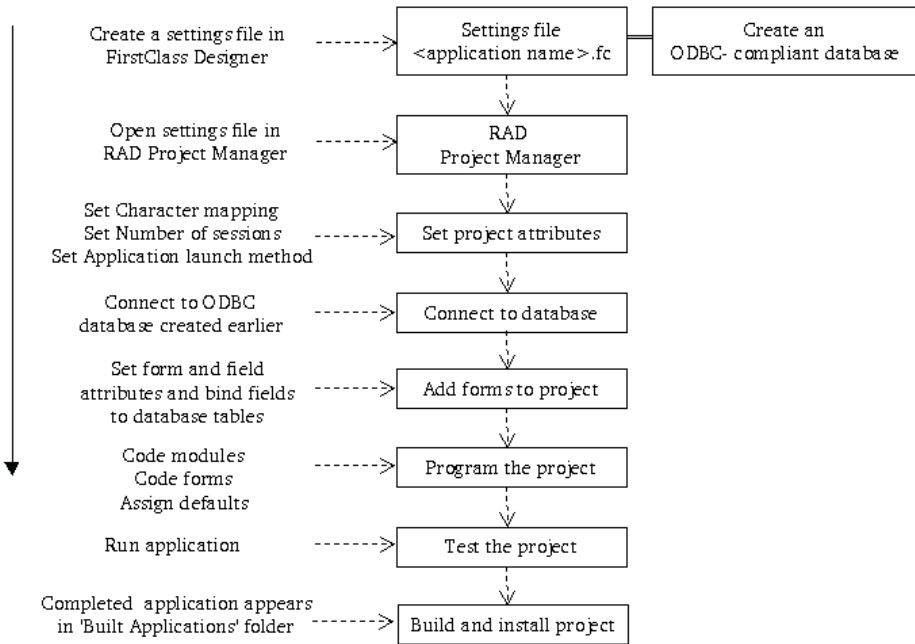
- creating your settings document
- creating your forms
- designing your forms
- adding your forms to your settings document.

### Flowcharts of FirstClass RAD applications

The following flowcharts present a graphical overview of how FirstClass RAD applications are created.

This flowchart shows the steps involved in creating a FirstClass RAD nondatabase application is created:

FirstClass RAD nondatabase application flowchart

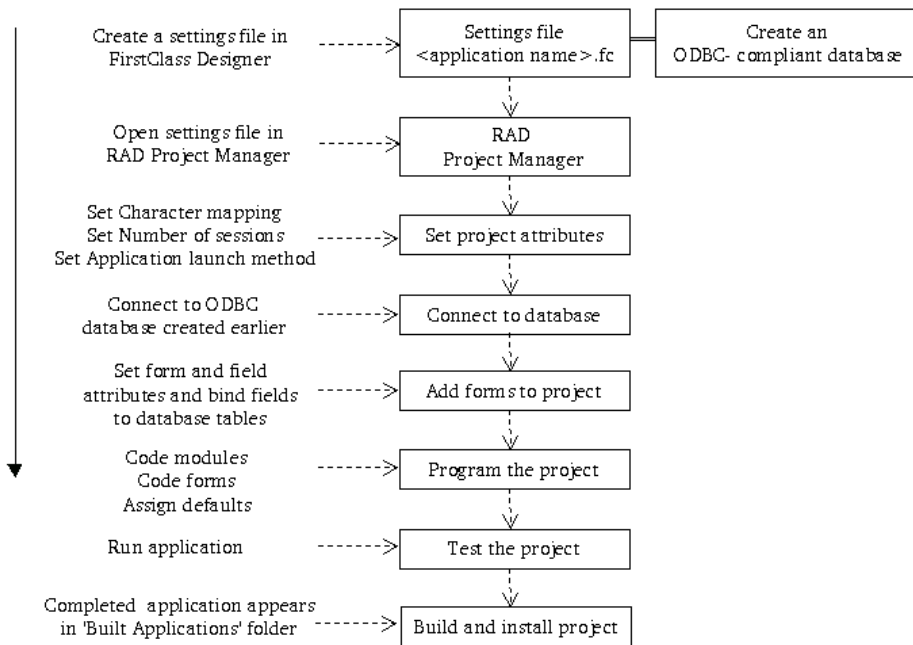


2

This flowchart shows the steps involved in creating a FirstClass RAD database application is created:

## Creating your FirstClass RAD applications

FirstClass RAD database application  
flowchart



### General steps for creating a FirstClass RAD application

This is a general overview of the steps involved in creating a FirstClass RAD application. For information on how to create an application see Chapter 7, "Creating your FirstClass RAD applications" and our online help.

1. *for database applications only*  
Create an ODBC data source for the database.
2. Create a settings file that contains all the resources (for example, custom icons and forms) your application will require, using FirstClass Designer.
3. Create the project and set its attributes.
4. *for database applications only*  
Connect to the database and its tables.
5. Add the forms to the project.
6. Name the fields on the forms.

7. *for bound column databases only*  
Link the fields on the form to the database columns.
8. Program the project.
9. Test the project.
10. Build the application.
11. Install the application in FirstClass.

Applications built in FirstClass RAD are automatically reloaded on the FirstClass server.

**Note** Applications run during the build will continue to use the older project build configuration. Applications run subsequent to the build will use the new FCX application template.

Applications built in FirstClass RAD automatically create their own stationery installations and functional application icons in the Built Applications folder. For information on creating stationery for your applications, see our online help.

**2**

## Maintaining your FirstClass RAD applications

After you have planned and created your FirstClass RAD applications, you need to maintain them. This is done through the FirstClass RAD Application Monitor, see “Maintaining FirstClass RAD” on page 55.





# FirstClass RAD concepts

There are basic concepts in FirstClass RAD you need to understand before creating your applications. These concepts include:

- managing projects
- using forms and resources
- databases
- source code.

## Managing projects

When starting a project, you create a FirstClass RAD application in the FirstClass RAD Project Manager, which provides access to:

- information about the project as a whole
- information about the forms in the project
- all code modules.

The Project Manager runs on the FirstClass server and is accessed from the FirstClass client. The Project Manager contains all of the components of any FirstClass RAD application including:

- forms
- code modules

A code module stores subroutines and declarations (variables, constants, and user-defined data types) that can be called from anywhere in the project (for example, from different forms or different fields on a form). This provides a centralized place to store code that is used multiple times.

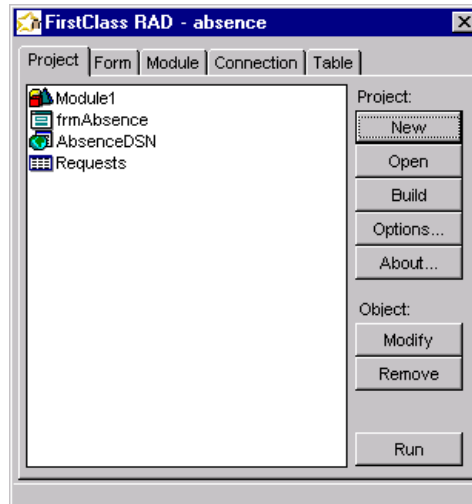
- database objects (if applicable).

The FirstClass RAD Project Manager lets you run your application in debug mode for testing purposes, without having to rebuild and reinstall your application every time you want to test it. The buttons on each tab in the Project Manager also lets you perform many local functions, for example creating, modifying, or removing tab specific items. For information on tab buttons, see our online forms help.

Let's take a look at the different tabs in the Project Manager.

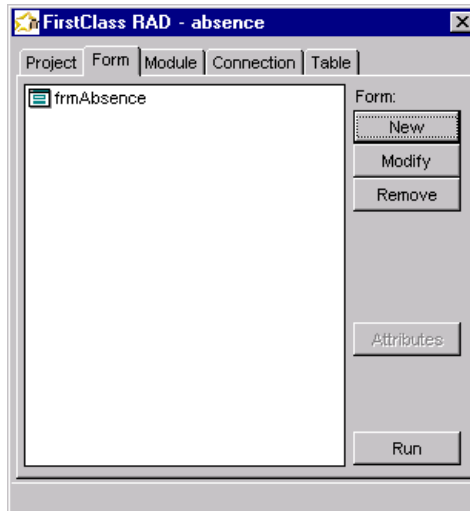
Project tab The Project tab displays all of components of your FirstClass 6 Rapid Application Developer project including:

- code module
- form(s)
- ODBC connection
- table(s).

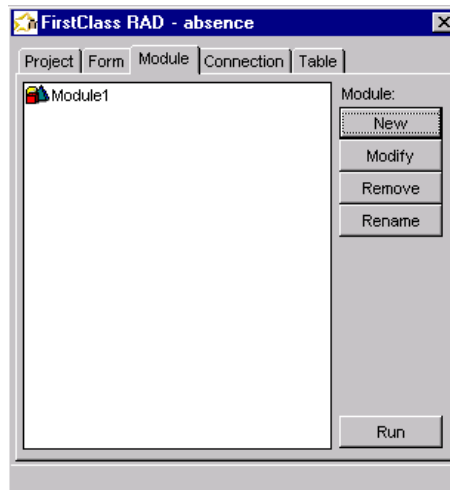


3

Forms tab The Forms tab displays the forms used in your FirstClass 6 Rapid Application Developer project.



Module tab The Module tab displays the modules used in your FirstClass 6 Rapid Application Developer project.

**3**

Connection tab The Connection tab is used for applications that connect to databases.

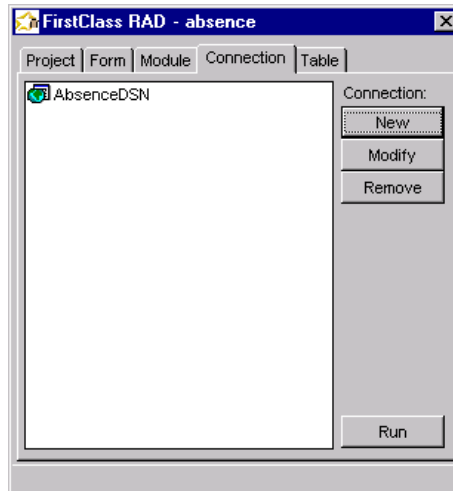
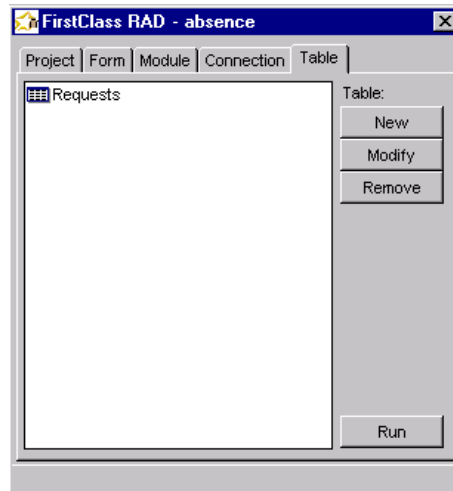


Table tab The Table tab displays the tables (if you are using a database) used in your FirstClass 6 Rapid Application Developer project.



3

## Using forms and resources

FirstClass resources, and the forms they contain, provide the visual interface for your FirstClass RAD applications. You create forms in FirstClass Designer and bring them into FirstClass RAD through the Forms window in the Project Manager (see "Forms

tab" on page 17 in this chapter). While in *FirstClass Designer*, you program the fields and buttons that you placed on these forms to do various tasks and events. In a database application, you would link these forms to ODBC-compliant databases to be able to pull relevant data into your application. For details on how to create forms, see *FirstClass Designer*.

## Working with databases

FirstClass RAD applications can connect to any ODBC-compliant (Open Database Connectivity) database whether as simple as a Microsoft Access or Excel file, or as vast as a large scale Oracle or Microsoft SQL database. ODBC is the industry standard for relational database access.

ODBC allows you to write programs that use SQL to access, update, and delete data contained in a backend database without knowing what type of database you are connecting to. This means you can move the data from a simple Access file to a larger SQL server at any point without breaking your application or rewriting large amounts of existing code or interface.

A database is accessed via an ODBC driver, which provides connectivity to a single type of database. Most major manufacturers of database products provide drivers for their products free of charge. Microsoft provides drivers for their products and a few others such as Oracle, text files, and comma-delimited text files. FirstClass RAD handles the ODBC connection, so all you have to do is set up one DSN (Data Source Name) on your Windows NT server and you can write programs that access that database from any location.

## How FirstClass RAD forms and databases work together

FirstClass RAD applications interact with the user in several different ways. The most basic is a simple console, to which you can print information. In order to make the application more useful and interactive, you must design forms using FirstClass Designer. These forms are just like any other FirstClass form, so you can automatically distribute them to the end user when they

run the application, or you can build them into FirstClass settings files, which can be distributed with the client settings or as a file attachment. For more information on creating and distributing forms and other FirstClass resources, see *FirstClass Designer*.

A FirstClass RAD form can tie into multiple database tables and multiple databases. The databases can also be on different servers. For fields that will always contain data from the same source (the same column in a database table), you can create a bound-column database application. This type of application ties fields to the appropriate database columns, providing a simple point-and-click interface that makes database information quickly accessible to users.

Examples of uses for bound databases are:

- displaying records where you want all information visible
- writing applications quickly with less coding.

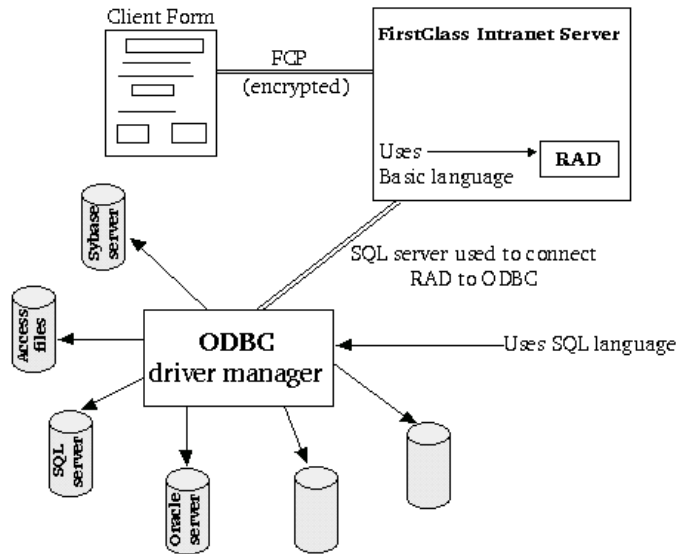
For more flexibility, when you do not have fields that are bound to a particular data source, you can create a non-bound database application.

Examples of uses for nonbound database access are:

- creating forms with fields that can contain data from various sources, such as a search results field
- developing applications that do not use forms to display data, such as an application that generates batch admin script.

Let's take a look at a diagram showing FirstClass RAD interaction with databases.

## How FirstClass RAD works with databases



3

Since FirstClass RAD uses the client interface to present information to the user, his experience is tied directly to the existing functionality of FirstClass. For example, you can access database access, electronic mail and conferencing, calendaring, and other features from a single client. Data processing is done on the server end, so access speed is fast and the application does not consume client resources.

As the database connects to the FirstClass server, the database itself remains safely behind a firewall and FirstClass permissions are used to restrict the user's access to the data. Any user can launch the FirstClass RAD development environment, as long as they have a copy of application and can design applications from any FirstClass client. The link between the user, developer, and server is fully encrypted and secure.

## Using source code

The source code in FirstClass RAD applications is the raw BASIC code that allows your program to function. Anyone with access

to that source code can make changes to the program and customize the program to do other things.

When you create a project in FirstClass RAD, the source code is stored on your server in the /FCPO/FCRAD/Source folder. This allows you to access the development environment from any computer with a FirstClass client and work on your code from that location.

## Distributing completed applications

When you build your project, the source code is parsed into an executable format and an FCX file is created in the server's /FCPO/FCRAD folder. When an application is compiled, that application is loaded. When the server is started, all applications are loaded. You can remove an application from this folder, if you don't want the application to run the next time you start your server.

**Note** If a previous version of the project's FCX file is in the FCRAD folder, it is overwritten with the newer version.

FCX files are encrypted to protect your source code from outside users. This means you can distribute a completed application to someone else without providing them with the code you used to create the program. Built projects automatically create their own stationery installations, which are located in the Built Applications folder. For information on creating different application stationery, see our online help.

## Branding your application

You should record all project information on the Get Info tab on the Application Information form. You access this tab from the Options button, located on the Project tab in the FirstClass 6 Rapid Application Developer Project Manager. For an explanation of the Application Information form fields, see our online form's help. For your records, you should use this form to brand your applications prior to distribution.



You can also view any application's information through the Get Info tab on the Application Monitor.

## Running FirstClass server applications

You can run FirstClass 6 Rapid Application Developer applications within the FirstClass server and without the context of a client login. These applications are commonly used as 'watcher' applications, which allow FirstClass RAD to constantly monitor the FirstClass server or databases and make changes as required. These applications are especially useful when used in conjunction with FirstClass server statistics events. For information on FirstClass servers and server statistics, see the administrator's section of our online help.

Server applications can be created to perform different functions, including:

- querying a personnel database every few minutes and automatically creating a new FirstClass account for all new employees who are entered
- retrieving server statistics event information (such as a new user created) and updating the database information
- creating and scheduling notifiers
- generating email messages at any time.

You create FirstClass Server applications just as you would create any application in FirstClass RAD. To enable an application to be run as a server application, the developer simply goes to the Project Options dialog in FirstClass RAD and sets the Launch popup to SERVER APP. You can also use the Sleep BASIC command to halt execution of a program until a specified time period has elapsed, or until a certain time or date has been reached. For information on the Sleep BASIC command, see our online help.

Server applications will ignore any FirstClass RAD code that cannot be run outside of the client login context (for example, showing forms).



# Coding concepts

FirstClass RAD uses object oriented programming to create the application interface, and uses the BASIC programming language to code the event procedures on forms.

This chapter explores standard FirstClass specific information, BASIC language concepts and their role in FirstClass RAD, specific FirstClass RAD programming features, and database application basics. For coding procedures, see our online help.

## FirstClass specific information

FirstClass RAD uses FirstClass programming commands specific to the FirstClass environment (client and server). They are

- FCBatchAdminCode
- FCBatchAdminReply
- FCEventShiftState
- FCClientPlatform
- FCClientVersion
- FCGetPrivGroup
- FCPrivGroup
- FCPOFolder
- FCServerSerialNumber
- FCUserCID
- FCUserFirstName
- FCUserID
- FCUserLastName

**4**

- FCUserMI
- FCUserName
- FCUserPrivGroup.

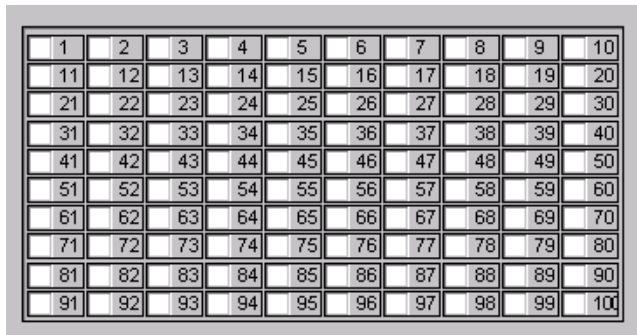
For a complete explanation of these commands, see our online help.

## Field arrays

FirstClass RAD allows the creation of field arrays. Field arrays allow FirstClass RAD developers to perform operations on groups of similar types of fields, utilizing an indexing method rather than explicitly referencing a unique name for each field. For steps on how to create a field array, see our online help.

Field arrays are particularly useful when assigning properties to large groups of controls.

Field array for 100 checkboxes



1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

To create a field array:

1. Open the form in FirstClass RAD and assign a name to a field
2. Assign the same name to a field of the same type on the form.

If you have completed these steps properly, you will then be prompted to create a field array. The field is assigned an index that will be visible in the Attributes window. Assigning the same name to additional fields on the form of the same type will include them in the field array.

Once all of the desired fields have been added to the field array, each field can then be referenced by its identifier in conjunction with its index. For example, the following code could be used to protect 100 checkboxes:

```
Dim i as integer
For i = 0 to 99
    cboMyCheckBoxArray(i).Protected = TRUE
Next
```

### Box arrays

Box arrays are accessed in an identical fashion to arrays; values can be retrieved or assigned using parentheses and an index. For example,

```
myBox(23) = "Hi there"
```

Before any manipulation of box arrays can occur, this list must be created. This is done by assigning values to each element of the box array in order starting from the first element 0. For example:

```
Dim i As Integer
For i = 0 to 99
    MyBoxOfStrings(i) = "String #" & Str(i)
Next i
```

Once the list is created, it cannot be collapsed. However, the list may be reassigned. If the reassigned list is shorter than the original list, there will be 'dangling list items'.

When a box array item gets focus, its index attribute is set. For example:

```
Sub Click()
    If MyBoxOfStrings.Index = 10, Then
        Print "You clicked on the 10th element in the list."
    End If
End Sub
```

Box arrays now include the Key attribute. In addition, to the standard data value associated with an entry in a box array, each entry may now store a second non-displaying data value. The data value stored in the Key attribute may be of any type. However, once assigned at run-time, the data type of the Key field for the box array cannot be changed. For example:

```
For i = 0 to 100
    BoxArray(i) = "Hi There!!"
    BoxArray(i).Key = "This is hidden Key #"; Str(i)
Next
```

### Fixed and expanding lists

Fixed and expanding can contain multiple items of data. A fixed list stays exactly the size that it was created as in <BookTitle>FirstClass Designer, no matter how many items are added to the list. If more items are added to the list field than can be displayed within the allotted space, a scroll bar is automatically provided in order to scroll up or down through the items in the list. An expanding list is the size of a single item. Each item that is added to the list increases the size of the field downwards to accommodate more data.

Both field types support user actions based on the entire field, but retrieving the Index attribute of the list allows you to determine which element within the list was selected by the user. For example, if a user clicks an item, the Index attribute is set to the item number so the user knows which item was clicked.

List elements can also have a key attribute that is unique to each item in the list. Individual list items can be assigned or retrieved as if the list is an array variable, ListName(0) is item number one from the list.

### Statistics events

FirstClass RAD applications respond to events that occur on the FirstClass server. By selecting the Server Statistic Events checkbox on the General tab, you expose the StatsEvent module.

To expose the StatsEvent module, in FirstClass RAD Project Manager:

1. Click the Options button on the Project tab.
2. Check the Server Statistics Events field on the General tab of the Project Attribute form.
3. Click OK.

The StatsEvent module contains a number of Subs that are called automatically when server events occur. Each method may be programmed to perform operations when the events occur.

Also included in the StatsEvent module is the EventMain subroutine. Just like a method, the EventMain subroutine is also called each time an event occurs on the FirstClass server. The order of operations occurs like this:

1. A FirstClass event occurs (Login, GetInfo, ...).
2. The EventMain subroutine runs.
3. Event-specific subroutine runs (StatsEvent\_Login, StatsEvent\_GetInfo,...).

For example, if you want to display a message on the console each time a new user is added to the system, you would program the appropriate method:

```
Sub StatsEvent_AddUser(theStatsEvent as StatsEvent)
    Console("Added a new user!")
End Sub
```

Each method receives as a parameter 'theStatsEvent' of object type StatsEvent. The StatsEvent object type contains data and methods that detail the particular event that has occurred and has the following methods and attributes:

<b>Timestamp</b>	a date data type that contains the date and time of the statistics event
<b>StatsEvent</b>	the FirstClass statistics event code
<b>Delimiter</b>	parsing delimiter attribute for the event text
<b>Text</b>	the text of the event as it would be written to the FirstClass server log file

<b>Text(n)</b>	the text of the 'n' column automatically parsed with the delimiter specified by the delimiter attribute
NumCols	the number of columns in the delimited text of the event string (using the delimiter attribute)

In order for FirstClass RAD to trigger any statistics events you must check each statistic event you want to log on the Statistics & Billing > Statistics and Billing Control form on the administrator's Desktop. Also, you must have a valid folder path entered in the Statistics folder path field to log these events in a log file. If you do not specify a valid folder path, the chosen events will not be triggered.

The following events are received by a FirstClass RAD application and may be programmed using FirstClass RAD:

AddUser	ApplyModel	Approve	Attach
AutoReg	ChangeUser	Close	CommLinkFail
ConfDelivery	ConflItemDel	ConfPermissions	ConfSubscribe
Control	Create	DeleteUser	Directory
Download	GetInfo	History	Login
Logout	MsgDelete	MsgDelivery	MsgForward
MsgReply	Open	OpenDeskTop	Password
Resume	SaveAttach	Search	Spam
Upload	Web		

## BASIC programming overview

The BASIC language provides you with the ability to create simple and useful applications. In FirstClass RAD, BASIC has been incorporated into a graphical interface, which drastically reduces the amount of coding that you need to do in order to create your programs.

The biggest advantage of BASIC is that it is an interpreted language and, therefore, is not compiled like many other languages. This means that a BASIC program can run immediately after any modifications to the code without having



to be compiled. When you run a BASIC program, the interpreter tells you when an error has occurred, and an error message appears on your screen. Interpreted languages also cannot "crash" but generate runtime errors that are not critical. When your code is correct your program performs as expected.

### **Object Oriented programming**

Object Oriented programming emphasizes the application's interface and the user's interaction with the interface. Basically, you draw the application interface in a graphical forms editor (FirstClass Designer), and then program the 'Event Procedures' of the objects or fields contained in the interface.

### **Event procedures**

Event procedures run when the user manipulates interface objects like fields and forms through clicks, double-clicks, changing focus, changing values, or other events. BASIC code commands may be added to any one of these event procedures. When the event procedure is triggered by the user, the corresponding code is run. For example, clicking on a field runs the field's Click event procedure.

4

## **Coding conventions**

All of the code you write in FirstClass RAD is created, edited, and removed in the Code window. Follow these coding rules to make your code clear and easy to use.

### **Combining multiple lines of code**

It is good form to limit your code to one command per line. However, if you want to place two or more commands on the same line, use a colon (:) character between the commands. For example:

```
txtMyText1.txt = "" : MyForm.Title = "New"
```

### **Continuing code on a new line**

You can break up a long command into multiple lines by using the line continuation character (\_). For example:

```
txtTotalSales = 1345.45 + 345.25 + 369.58 + _  
2356.56 + 345.90
```

If you break a character string into multiple lines, you must use a matching set of double quotes (") and, to concatenate the lines, use the ampersand symbol (&). For example:

```
txtMyText = "This is the first half and " _
& "this is the second half."
```

## BASIC programming and FirstClass RAD

FirstClass RAD uses standard BASIC modules in its application development. Basically, modules are collections of procedures. There are two fundamental types of modules available in FirstClass RAD: Form modules and Code modules.

### Form modules

Form modules are an essential part of almost any application you create in FirstClass RAD. They contain all of the procedures that are assigned to a given form and the fields that the form contains.

FirstClass RAD allows you to write procedures (blocks of code) which run when a user triggers any of an array of events for a given field or the form itself. These event-driven procedures are called Event Procedures. See "Event procedures" on page 31. Each type of field has a variety of Event Procedures that can execute BASIC code. For instance, you can program the Click() procedure of a button to open a new window or you can program the Change() method of a text field, which runs when the information contained in the field changes, to validate new information.

Simple FirstClass RAD applications may use only a single form module. However, as a project grows, it may become necessary to add additional forms. If there are procedures used in more than one form, or are called from different fields in the same form, it may become necessary to put these procedures in a code module.

### Code modules

Code modules contain procedures that are global, or can be accessed by any of the objects in the entire project. Code modules are useful for creating procedures and declaring variables that are used throughout the project. For example, you can store one copy of a procedure in a code module that can be called from any other procedure in the project with a single line

of code. This simplifies and streamlines projects by eliminating copies of identical code.

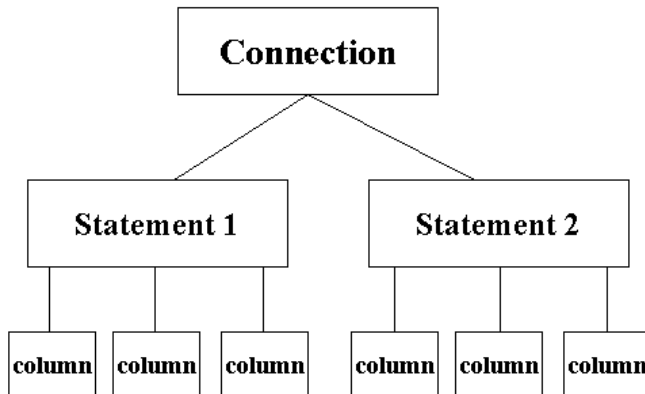
As an added benefit, when a procedure is modified, its changes are reflected every place the procedure is called. Duplicated code must be modified at each occurrence, which can lead to program errors. For steps on programming, see our online help.

## Database application basics

ODBC is a platform independent database access standard that has been adopted by the majority of database companies on virtually every computer platform. FirstClass RAD uses ODBC in Windows NT to connect to databases. FirstClass RAD makes it possible to create any database interface with its bound tables database interface and object oriented database language.

Object oriented programming syntax is designed to organize its Application Programming Interface (API) into an accessible syntax that organizes all database objects in a simple three-level hierarchy: Connection, Statement, Column.

Database object hierarchy



Each of these objects has attributes and methods that are used to manipulate the object.

**Connections**

Connections act as conduits through which all database statements travel.

*Connection objects*

The attributes and methods of a Connection object are used to:

- establish the connection

This includes passing login and password information to the database driver.

- set connection options.

This includes transaction isolation, cursor type, timeouts, and tracing report information about the database driver.

**Statements**

Statements are responsible for executing database operations on a connection. A statement specifies all aspects of a database operation including the SQL query, resulting records, and all of the attributes of the returned cursor.

*Statement objects*

The attributes and methods of a Statement object are used to:

- execute SQL queries
- store data that is returned as a result of a query (data is stored in a temporary set of columns and rows called a cursor or result set)
- manipulate data in the cursor (rows in the cursor can be browsed, changed, deleted, and added to)
- set and retrieve statement attributes, including information about cursor types, concurrency settings, and maximum number of rows.

**Columns**

Columns define the column of data that is stored in the cursor, when a query execution returns data. The Column object makes it possible to specify a single column or set of columns and the rows of data contained within them.

For examples of Connections, Statements, and Columns, see our online help.

*Column objects*

The attributes of a Column object are used to:

- specify a column or set of columns in a cursor
- store the attributes, such as Text, Value, Precision, Name, and DataType, of a column
- change database values.



# Installing FirstClass RAD and logging in

This chapter describes the system requirements and installation procedures for FirstClass Rapid Application Developer 2 for Windows 95, Windows 98, Windows 2000, Windows NT, and Windows® XP.

## Windows

You can install FirstClass RAD on Microsoft Windows 95, Windows 98, Windows NT, Windows 2000, or Windows® XP. FirstClass RAD can also be installed on FirstClass Personal for demonstration and offline development with two session licenses. This means you can only run two FirstClass RAD applications simultaneously (use only two session licenses).

### System requirements

To install FirstClass RAD, you must have the following already installed on your Windows machine:

- FirstClass client
- FirstClass server 5.5 Gold or higher and a working FirstClass post office or FirstClass Personal.

**Note** FirstClass version 7.0 is required for some functionality, such as single tab support for 255 characters, and greater character batch administration replies.

### Installing FirstClass RAD

To install FirstClass RAD on your system:

1. Search the FirstClass® Rapid Application Developer CD for the installation folder.

2. Double-click the FCRAD application icon.



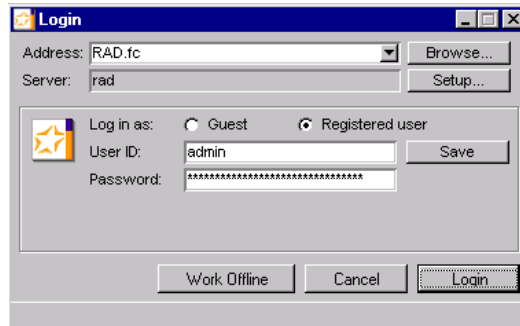
3. Follow the prompts.

## Mac® OS

FirstClass RAD cannot be installed on a Mac OS machine, however you can develop on a Mac OS client.

## Logging in to FirstClass RAD

To open FirstClass RAD, start your FirstClass server and log into FirstClass RAD as the administrator using the FirstClass client:





# **Planning and creating FirstClass RAD applications**



# Planning your FirstClass RAD applications

In "Planning your FirstClass RAD applications" on page 9, we introduced the application planning stage. In this chapter we build on those concepts.

## Planning questions

There are a number of questions you should ask yourself before starting to create your application. The most important questions include:

- What task(s) must the application perform?
- Will a database be included in the application?
- How are you going to display the data to users?
- What kind of forms do you require? Will they include graphical items such as icons and pictures?
- How will the application run? Will the application run on the server or launched by users when they click an icon?
- How will you distribute the application to your users? Will you email the application as an attachment or place it in a conference for general accessibility?

**Note** You must create stationery for any application you put in a conference or a folder.

After you have answered these questions, you can start gathering the necessary resources for your application. For example, if you decide to use a database, you must make sure that it is ODBC-compliant, see "Working with databases" on page 19 for more information. If you require forms in your application, you must create these forms in FirstClass Designer.

In the next chapter, "Creating your FirstClass RAD applications", we build on the concepts introduced in "Creating your FirstClass RAD applications" on page 10.

# Creating your FirstClass RAD applications

This chapter details the steps involved in creating your settings document, resources, and FirstClass RAD applications to complete a FirstClass RAD project. For further information on programming projects and the steps involved, see our online help. The FirstClass RAD language reference section of the online help provides descriptions of the commands, statements, and variables you can use when creating your applications. Also, to work through real FirstClass RAD applications, see "Tutorials" on page 61.

## 7

### Creating your FirstClass settings file and resources

As introduced in "Planning your FirstClass RAD applications" on page 9, you must first create your settings document and resources (forms) you will use in your FirstClass RAD project.

#### Creating your settings document

The first step in creating your resources is to create a settings file that contains all of the resources (for example, custom icons and forms) your FirstClass RAD application will require. You must use FirstClass Designer to create this settings file. For detailed information on creating settings files, see <BookTitle>FirstClass Designer and our online help. You may want to give this settings file the same name as your application. For example, <application name>.fc (the settings file must have the .fc extension).

FirstClass RAD requires that you follow certain conventions when building this settings file:

- all forms must be Database Query forms, with IDs in the 8000–8999 range or Database Hitlist forms, with IDs in the 10000–10999 range
- all forms must be templates, unless you need the form accessible as stationery through the Admin > New Stationery menu
- form IDs must be unique and unchanging  
Once you add a form to a project, you cannot change its form ID.
- all field IDs on the forms must be unique and numbered 1000 or higher.

### **Creating your forms**

Now that you have created your settings document, you must create the forms it will contain. Forms are also created in FirstClass Designer.

Creating forms is a matter of drawing fields on the form and adding the resources, such as icons, pictures, and even sounds as required. For excellent examples of how forms should be prepared, open the FirstClass settings document included with the sample materials that ship with FirstClass Designer. The document is called Example.fc. When preparing your settings documents, remember the conventions outlined in "Creating your settings document" above.

Once your settings are prepared, you use FirstClass Designer to create the forms that will be used for your FirstClass RAD project.

### **Designing your forms**

FirstClass Designer lets you to create new form resources and add fields to these forms. For detailed information on creating forms, see <BookTitle>FirstClass Designer. FirstClass RAD has some basic requirements for resources that are included in the settings document:

- field IDs on the forms must be unique and numbered 1000 or higher (this is the default)
- field IDs must be unique and unchanging

Once the form has been opened in FirstClass RAD, you should not change any of the field ID numbers.

- use any type of fields on the form with the exception of 'Message Fields', which are designed to only work with FirstClass Mail forms.

**Adding a new form to your settings document**

When you add a new form to your settings document, you must follow these steps to make sure the form will be enable to work with FirstClass RAD:

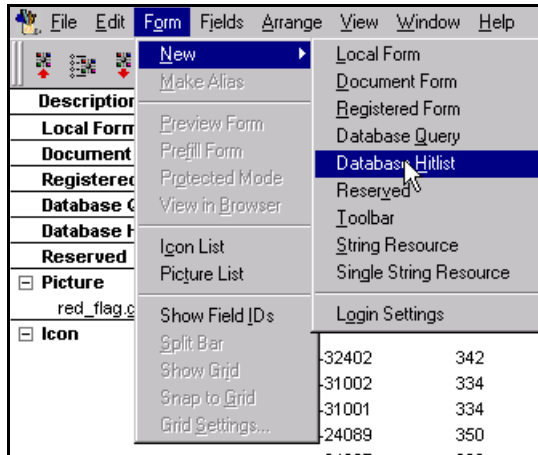
1. Open your settings file with FirstClass Designer.

This is the file you initially created, for example <application name>.fc. See "Creating your settings document" on page 43.

2. Select Form > New, from the pulldown menu and choose either Database Query or Database Hitlist from the list.

A new form will open.

Designer form - Database Hitlist



3. Select Form > Split Bar from the pulldown menu to remove the split bar from the form.

**Note** FirstClass RAD forms are not currently able to use the text editor under the split bar.

4. Close the form and choose Save. You will then be prompted to enter the form's properties.

Form properties

The screenshot shows a 'Properties' dialog box with the following fields and values:

- Type: Form Template
- Name: (empty)
- Title: Untitled Form
- Range: Database Hit List
- ID: 10000 (with a 'Unique ID' button)
- Icon: (empty)
- Style: Standard
- Menu ID: Default Menu
- Character set: Macintosh Roman
- Form position: Top: 0, Bottom: 245, Height: 245, Left: 0, Right: 310, Width: 310

5. Input a Name and Title for your form.
6. Input a unique form ID.

**Note** Make sure this ID does not overlap with other form IDs in your current production settings file or with the form ID used in the RAD.fc settings file that comes with FirstClass RAD.

7. Click OK on the properties window to save your changes.

### Drawing fields on your form

Now that you have created your form, you must add the fields. It is recommended that you complete the design of your form before adding it to your FirstClass RAD project. Keep in mind, you may add fields to forms after you have added the form to your FirstClass RAD project. However, we discourage removing fields.



**Remember** FirstClass RAD may use all of the field types in FirstClass Designer except Messaging Fields.

To add a new field:

1. Open the form.
2. Select the Fields menu and select the desired field type from the list of field categories.

Designer field categories



3. With your mouse pointer, click on the desired location for the field and draw a box.
4. Adjust the size or location of the field, if necessary.

Repeat these steps for each field you wish to add. Save and close your form when you are finished.

For a table listing of all the field types, field methods, and their corresponding field attributes as they apply in FirstClass Designer, see "FirstClass Designer field attributes" on page 145 in the Appendix.

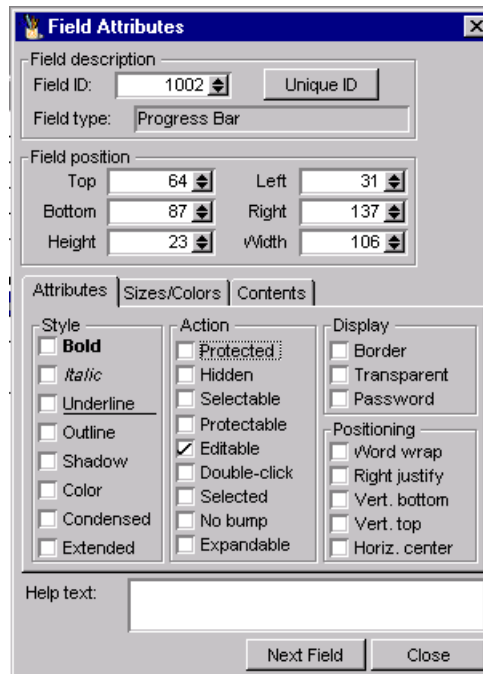
### Setting field attributes

After you have added your fields to your form, you need to set your field attributes. To do this, you use the Attributes window. Attributes include style elements (hidden, protected, bold, italic, and so on), position (top, left, height, width, and so on), size/color and contents (text, lists, and so on).

To add field attributes:

1. Open the form
2. Select Fields > Field Attributes from the Fields pulldown menu.

Designer field attributes



3. Choose a field on your form to make the field attributes window active.
4. Save and close your form when you are finished.

## Working in the Project Manager

Now that you have finished creating your resources for your FirstClass RAD application, you must bring these resources into a FirstClass RAD project and create and complete your desired application.

In Chapter 3, "FirstClass RAD concepts", we introduced the basics of the Project Manager and its interface. Please reference this chapter if you are unsure about any of the components of the Project Manager.

In Chapter 2, "Administrator tasks", we introduced the different flow of events and general steps for creating different types of FirstClass RAD applications (nondatabase, bound database, and nonbound database). In the next section, we provide step-by-step instruction for pulling together all of the different components of your FirstClass RAD project and completing the FirstClass RAD application itself.

## Creating a nondatabase application

# 7

To create an application that does not connect to a database:

1. Create your settings file, as described in "Creating your FirstClass settings file and resources" on page 43.
2. Use FirstClass RAD Project Manager to create a project for this application.

**Note** For further information on working in the FirstClass RAD Project Manager, see Chapter 3, "FirstClass RAD concepts".

3. Set project attributes.

Attributes that apply to the project as a whole include:

- how the application will be launched
- whether users can run multiple concurrent sessions of the application

- the total number of concurrent sessions of the application that can be run on the FirstClass server.
4. Add your forms to the project.
  5. Set the following form attributes:
    - the form's name
    - the title that will appear in the form's title bar
  6. Name the fields on each form.
  7. Program the project.

This involves:

- adding code modules if you want to use separate modules to organize the project (by default, there is one code module)
- writing code for the forms
- validating content in form fields
- assigning defaults.

**Note** You may have done some of the work in the last two points when you created the forms in FirstClass Designer.

8. Test the project.
9. Build the application.
10. Install the application in FirstClass.

This requires distributing the necessary resources to your users. Resource distribution is discussed in <BookTitle>FirstClass Designer.

## Creating a bound-column database application

To create a bound-column database application, you follow the same steps but with the inclusion of a database. To create a database application with bound columns:

1. Create an ODBC data source for the database.
2. Create your settings file, as described in "Creating your FirstClass settings file and resources" on page 43.
3. Use FirstClass RAD Project Manager to create a project for this application.

**Note** For further information on working in the FirstClass RAD Project Manager, see Chapter 3, "FirstClass RAD concepts".

4. Set project attributes.

Attributes that apply to the project as a whole include:

- how the application will be launched
- whether users can run multiple concurrent sessions of the application
- the total number of concurrent sessions of the application that can be run on the FirstClass server.

5. Connect to the database and its tables.

6. Add your forms to the project.

7. Set the following form attributes:

- the form's name
- the title that will appear in the form's title bar
- the database tables to which fields on the form will be bound.

8. Set the fields' attributes for each form.

Field attributes include:

- the field's name
- *if this field will be bound to a database*  
the database table and column to which the field is to be bound.

9. Program the project.

This involves:

- adding code modules if you want to use separate modules to organize the project (by default, there is one code module)
- writing code for the forms
- validating content in form fields
- assigning defaults.

**Note** You may have done some of the work in the last two points when you created the forms in FirstClass Designer.

10. Test the project.
11. Build the application.
12. Install the application in FirstClass.

This requires distributing the necessary resources to your users. Resource distribution is discussed in <BookTitle>FirstClass Designer.

## Creating a nonbound- column database application

To create a nonbound-columns database application with non-bound columns:

1. Create an ODBC data source for the database.
2. Create your settings file, as described in "Creating your FirstClass settings file and resources" on page 43.
3. Use FirstClass RAD Project Manager to create a project for this application.

**Note** For further information on working in FirstClass RAD Project Manager, see Chapter 3, "FirstClass RAD concepts".

4. Set project attributes.

Attributes that apply to the project as a whole include:

- how the application will be launched
  - whether users can run multiple concurrent sessions of the application
  - the total number of concurrent sessions of the application that can be run on the FirstClass server.
5. Connect to the database and its tables.
  6. Add your forms to the project.
  7. Set the following form attributes:
    - the form's name
    - the title that will appear in the form's title bar
  8. Name the fields on each form.
  9. Program the project.

This involves:

    - adding code modules if you want to use separate modules to organize the project (by default, there is one code module)
    - writing code for the forms
    - validating content in form fields
    - assigning defaults.

**Note** You may have done some of the work in the last two points when you created the forms in FirstClass Designer.
  10. Test the project.
  11. Build the application.
  12. Install the application in FirstClass.

This requires distributing the necessary resources to your users. Resource distribution is discussed in <BookTitle>FirstClass Designer.

## Recording application information

The Application Information form shows the following data entered by the application developer:

- version
- build number
- status
- file name
- developer information
- company information
- the list of people using the application
- the users who are currently running the application.

FirstClass Application Monitor



This form can be filled in when developing a FirstClass RAD application by clicking the Options button in the FirstClass RAD development environment.



# **Maintaining FirstClass RAD**



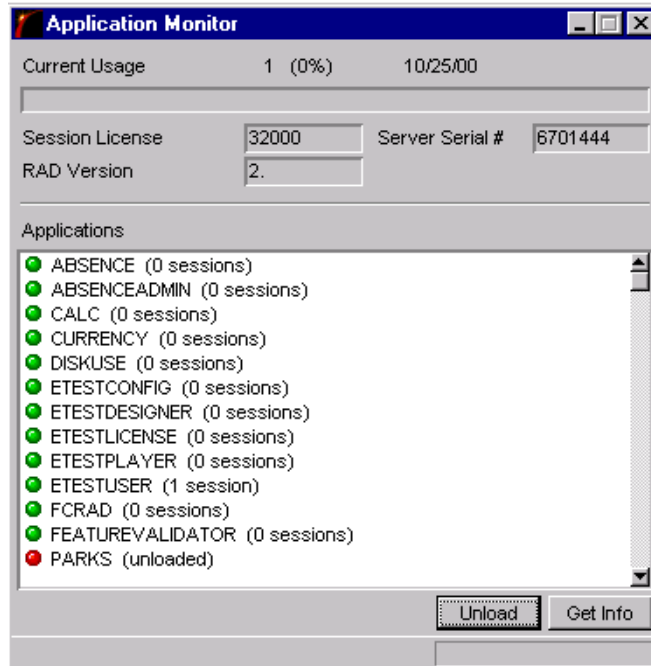
## Monitoring FirstClass RAD applications

The FirstClass RAD Application Monitor lets a you:

- see what applications are running on the FirstClass server
- manipulate applications
- load and unload applications on the server.

For a description of the Application Monitor form fields, see our [online forms help](#).

The monitor is located in the Application Monitor folder on the FirstClass RAD Desktop. When you double-click the icon, you should see this:



Any FCX file present in the /FCPO/FCRAD folder will be shown in the Application Monitor main window. A green light next to the application name indicates that the application is loaded and ready. A yellow light indicates that the application is being loaded or unloaded. A red light indicates that the application has not been loaded yet or has been unloaded.

Next to each application, there is a number indicating the number of open sessions for that application. Also, the version of FirstClass RAD, FirstClass Server serial number, and available FirstClass RAD licenses are displayed on this form with a bar indicating the total current usage out of the available licensed application sessions.

### **Loading and unloading an application**

When you compile an application or start the server, all FCX files on your system are loaded. You can view which applications are running from the Application Monitor. To unload or load an application from the server, select the application by highlighting the name and clicking the Unload button. The

indicator next the application is green for loaded, black for unloaded, and amber for in progress.

Loading an application will make it immediately available on the server. This means you can build an application on a separate server and put it online on your live server without having to shut it down. You can also revise an application, unload the old copy, and load the new copy without any downtime.

Unloading an application with no active sessions will happen immediately. Anyone trying to use the unloaded application will get an error message saying "Application not loaded on server". If there are active sessions, the application will not be unloaded until all users have quit the application. Any users trying to launch the application during the unload process will get the "not loaded" error message.

To display the Application Information form, double-click an application name (also accessible via the Get Info button).



# Tutorials





## Tutorial 1: Disk Usage utility (nondatabase)

Using FirstClass Batch Administration in conjunction with FirstClass RAD gives the developer the ability to deliver specific administrative powers to selected users without requiring full administrator or even subadministrator permissions.

In this tutorial, you will use the FirstClass Designer and FirstClass Batch Administrator to create an application to retrieve disk space quota and current disk usage information. You will also create a basic application interface in FirstClass Designer. The Disk Usage utility will have two data fields and two buttons. The two data fields will be used to display the disk space limit and the current disk space used. The Refresh button will be used to refresh the data fields with any disk usage changes that have occurred since the application was last run. The Exit button will close the form and exit the application.



As you work through this tutorial, complete every step in each section in order before moving on to the next section.

### Tutorial materials

To complete the Disk Usage utility project you need to have FirstClass Designer and FirstClass 6 Rapid Application Developer

loaded onto your system. Although some basic familiarity with FirstClass Designer is recommended, it is not required.

## Assembling the project

In this section, you will create and configure a new application form and cover the basics of using FirstClass Designer with FirstClass RAD.

### Adding a new form to your settings document

To add a new form to your settings document, locate your FirstClass RAD settings document /FCPO/FCRAD/Rez/RAD.fc and follow these steps:

1. Open your RAD.fc settings document with FirstClass Designer.
2. Choose Form > New > Database Hit List.  
The newly created form opens.
3. Choose Form > Split Bar.  
This will remove the split bar from the form.
4. Close the form and click Save.  
You will be prompted to enter the form's properties in the Properties window.
5. Set the following properties for the form:
  - Type: Form Template
  - Name: Disk Usage Tutorial Form
  - Title: Current Disk Usage
  - Range: Database Hitlist
  - ID: Either select a unique form ID in the numeric range between 10000-10999, or choose the default number.
  - Style: Nonresizable
  - Menu ID: No Menu
6. Click OK on the properties window to save your changes.

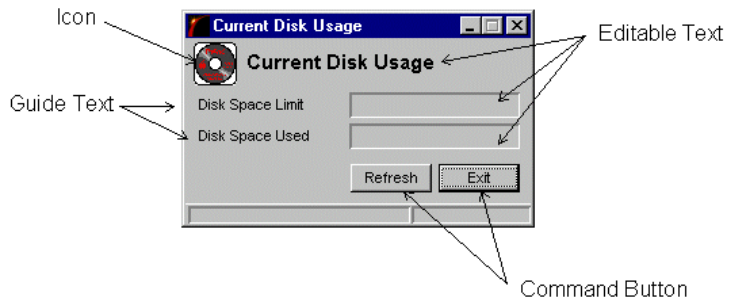
## Adding fields to your form

Now, you will add fields to your new form.

To add fields to your form:

1. Open your new form by double-clicking the form name in FirstClass Designer's resource list.
2. Select the Fields menu and select the desired field type from the list of field categories.
3. With the mouse, select the desired location for the field on your form and click. The field will appear.
4. Adjust the size and location of the field using the mouse and keyboard.

Repeat steps 2 - 4 and create the following application interface:



## Setting field attributes

Use the Attributes window to manipulate each field attribute. Attributes include style, position, and default elements, such as Hidden, Protected, Bold, and Color. For details on FirstClass Designer field attributes, see "FirstClass Designer field attributes" on page 129, in the Appendix and *FirstClass 6 Designer*.

If the Attributes window is not visible in FirstClass Designer, you can open it by selecting the Fields menu and choosing Field Attributes.

## Creating a new project

### Assigning attributes

Use the Attributes window to set the following attributes for each field.



Once the form has been completed, close your form, save your changes, and exit FirstClass Designer. You are now ready to log into your FirstClass application server and add the form to a FirstClass RAD project.

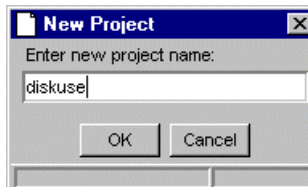
## Creating a new project

Log into your FirstClass server as the administrator. Then, run FirstClass RAD by double-clicking on its icon. The FirstClass RAD icon is located in the folder:

FirstClass RAD

To create a new project in FirstClass RAD:

1. Click New on the Project tab of the FirstClass RAD interface.
2. Type in a new project name not exceeding twenty-three (23) characters, and not containing any spaces. For this project, type 'diskuse'.



3. Click OK to create the new project.

The new project has been created. FirstClass RAD is now ready for you to begin programming your application.

## Working with forms and fields

The first step in preparing your new project is adding forms and naming the fields on the forms.

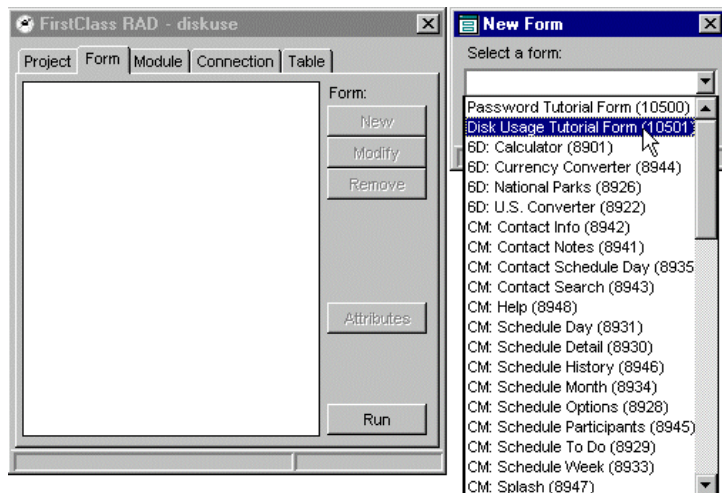
### Adding a form to your project

Adding the Disk Usage Tutorial form opens the form in FirstClass RAD. When you open the form, FirstClass RAD downloads all of the form information into the project database. Once you add a form, you can name and program all of its fields.

To add a new form:

1. Click New on the Forms tab.
2. Select Disk Usage Tutorial Form from the list of form in the settings file.

If you do not see the form in the list, make sure you are logged into the FirstClass server with the correct settings document.



3. Click OK to add the form.

The form is opened and the form's information is added to the project database. Also, the FirstClass RAD Attributes window will open to allow field naming and programming.

**Naming forms and fields** The next step is to name your form and the fields on the form.

To name the form:

1. *to display the default name of the form, the form title, and the form's bound data environment*

Click the Attributes window's Form tab.

2. Change the form's default name to 'frmDiskUse'.

The identifier frmDiskUse is what you will use to reference the form when you are writing code.

**Note** In FirstClass RAD you commonly assign a prefix to indicate an object's type. In this example, frm indicates this is a form object.

3. Press <Tab> to update the name change.
4. Enter the form's title as 'Current Disk Usage'.

The value of the title field in FirstClass RAD will be assigned to the title bar on the form when the application is run. This value overrides any values you have set in the FirstClass Designer.



5. Press <Tab> to update the form title change.
6. Move to the Naming fields section, without closing the form or the Attributes window.

If you have accidentally closed the form, simply reopen it by selecting it from the Form tab and clicking Modify. If you have accidentally closed the Attributes window, reopen it by clicking Attributes on the Form tab.

It is important to name the fields on the form that you intend to reference in your code; otherwise your code will not work. FirstClass RAD assigns a default name for each field if you do not assign one. However, these names are generally not specific enough to make your code easy to read. When assigning field names it is a good idea to assign unique values that are easy to remember.

In this application, you will only program editable text fields and buttons.

To name fields on the form:

1. Select the Field tab of the attributes window.
2. Click on the Editable Text field immediately to the right of the guide text 'Disk Space Limit'. The Attributes window will display the default name of the field.
3. Change the default name of the field from its current (StringField1001) to 'txtLimit', by modifying its name in the name field of the Attributes window.

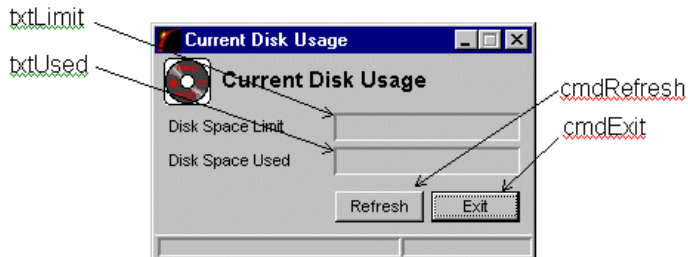
Again, you assign a prefix to indicate an object's type. Here, 'txt' indicates a text field object.



4. Press <Tab> to update the field name change.

Complete steps 2–4 of the field naming process and name the form's fields according to the following diagram. Don't forget to

press <Tab> after assigning the name of each field. When you have named all of the fields, close the form by clicking the form's close box.



## Programming your project

All applications begin running in the Sub Main subroutine in Module1. When your application runs, your startup form is shown from this code.

### Showing the form with Sub Main()

To open the Disk Usage utility form at application startup time, you will add one line of BASIC code to Sub Main().

To show the form with Sub Main():

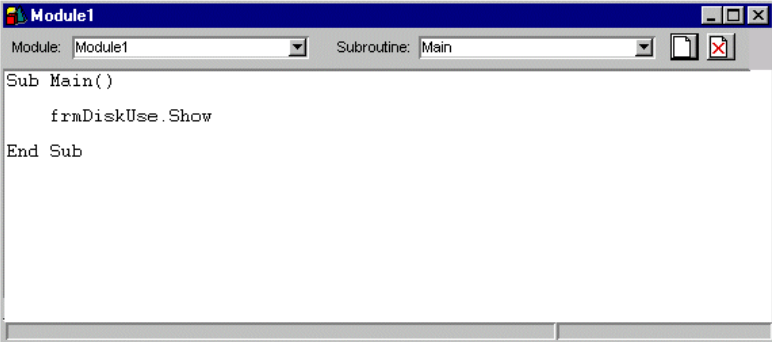
1. Select the Module tab in FirstClass RAD.
2. Select Module1 and click Modify. The Module1 code window will open.
3. Select 'Main' from the subroutine selection list.
4. Between the "Sub Main()" and the "End Sub" commands, type the following code:

```
frmDiskUse.Show
```

**Note** Always press <Tab> and close the Module window to save your code changes. If you do not, your code changes will be lost when you run your application.



Your completed Sub Main window should look like this:



```

Module1
Subroutine: Main
Sub Main()
    frmDiskUse.Show
End Sub

```

When you have finished typing in the code, press <Tab> and close the code window. Now that you have programmed Sub Main, the application is ready to run and test.

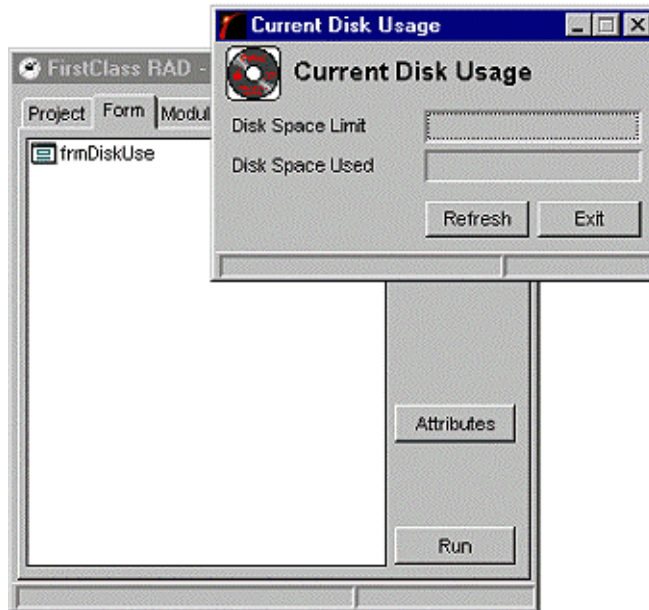
### Running the Disk Usage utility in FirstClass RAD

At this point, running the Disk Usage utility simply opens the form interface. Once other fields on the form have been programmed, the interface will respond intelligently to user interaction. For now, let's show the form in Debug Mode by running the application in FirstClass RAD.

To do this:

1. Click the Run button in the lower right hand corner of FirstClass RAD. The Disk Usage utility interface will appear. The Run button caption changes to 'Stop'.

**Note** If the Disk Usage utility interface does not appear, go back and check your steps in this section.



Your application is now being run in Debug Mode. Debug Mode applications run from within the FirstClass RAD environment just as if they were built and deployed on your system. Debug Mode allows you to test the application during development without fully installing the application on your server.

When you are finished testing your application, click 'Stop' to stop the running program and put FirstClass RAD back into Development Mode.

## Retrieving disk use information

It's now time to make our application get the user's disk usage information. First, let's consider the Batch Admin code used to retrieve a user's disk usage limit. The Batch Admin command to retrieve this information for the user JDOE would be the following:

```
GET USER JDOE 1357
```

Our challenge is to generate a similar Batch Admin command for any user running the application. The obvious problem, of course, is that you don't know the current user's User ID when you write your code. However, FirstClass RAD provides the `fcUserID` internal function to return the User ID of the current user running the application.

The FirstClass RAD command needed to retrieve disk usage limits from our application is:

```
BatchAdmin("GET USER " & fcUserID & " 1357")
```

What is happening here? You are supplying a compound text string as an argument to the Batch Administration command. Now let us look at how this string is formed.

### Forming compound strings

The argument you create for the Batch Administration command is made up of three distinct parts that are concatenated together with the ampersand ('&') operator. FirstClass RAD uses the ampersand operator to combine two or more text strings to form a single string. For example:

```
"Hi " & "There"
```

Evaluates to:

```
"Hi There"
```

In your Batch Administration command, you are combining three text values to create a single argument. The major difference between our command and the "Hi There" example is that not only are you concatenating literal text values, like "GET USER", but you are also concatenating the variable value of `fcUserID`.

The FirstClass RAD internal function `fcUserID` contains the User ID of the current application user. Let us do a test case; begin with the code in our application:

```
BatchAdmin("GET USER " & fcUserID & " 1357")
```

If you imagine that the current application user is 'JDOE' then, after FirstClass RAD does the `fcUserID` substitution, you have the following evaluation to perform in your program:

```
BatchAdmin("GET USER " & "JDOE" & " 1357")
```

Then FirstClass RAD evaluates all of the ampersand operators, and you are left with exactly the text string you want.

```
BatchAdmin("GET USER JDOE 1357")
```

The BatchAdmin then passes our compound string to the FirstClass Batch Administrator and the disk limit information is retrieved. How do you use this Batch Administration command in your program? In the following steps you will create a new subroutine to run your code.

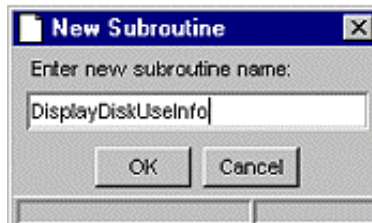
### Creating a subroutine

In the Disk Use utility, you will be retrieving the disk usage information from two events in your code: when the application is first run, and when the user clicks the Refresh button. Any time you expect to be running the same code from more than one event, you should create a new subroutine.

A subroutine provides you with the ability to centralize code, rather than duplicating it for each event. Having code in one place is essential for simplifying changes and preventing bugs. If you need to modify the code in a subroutine, you will only need to make the change in a single place. If your code is duplicated, you may need to make changes in two, three or even a hundred different places.

To create the DisplayDiskUseInfo subroutine:

1. Select the Module tab in FirstClass RAD.
2. Select Module1 and click Modify. The Module1 code window will open.
3. Click the new subroutine icon (blank page icon) in the upper right corner of the code window. The New Subroutine form opens.
4. Type DisplayDiskUseInfo to name the new subroutine.



5. Click OK.

The Module 1 code window now displays a new subroutine for you to program.

6. Between the "Sub DisplayDiskUseInfo()" and "End Sub" commands, type the following code:

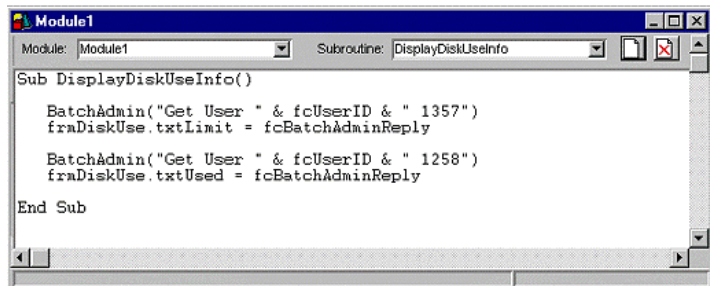
```
BatchAdmin ("Get User " & fcUserID & " 1357")
frmDiskUse.txtLimit = fcBatchAdminReply

BatchAdmin ("Get User " & fcUserID & " 1258")
frmDiskUse.txtUsed = fcBatchAdminReply
```

The four lines of code retrieve the disk use limit and disk use total for the current user and assign these values to the txtLimit and txtUsed fields on the frmDiskUse form. The fcBatchAdminReply function retrieves information retrieved from the last Batch Administration command executed and is used to assign values to your text fields.

**Note** When you specify a field name in a subroutine, you should always include the form name. This is done because an application may have many forms and a field reference is ambiguous if you do not specify the form name.

Your completed DisplayDiskUseInfo subroutine should look like this:



When you have finished typing in the code, press <Tab> and close the code window.

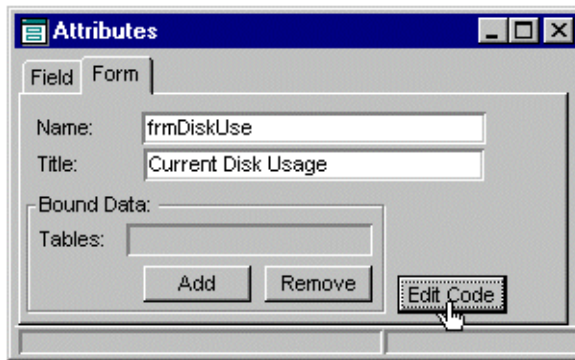
**Calling your subroutine**

Now that you have written the subroutine to retrieve the information, you will need to call it from somewhere in your application's code. The first time you want to retrieve this information is when the form is first shown. To do this, you will program the form's Load Event.

The Load Event on a form is run whenever a form is loaded or shown in your application. It is a good place to put form-specific initialization code.

To program the Load Event:

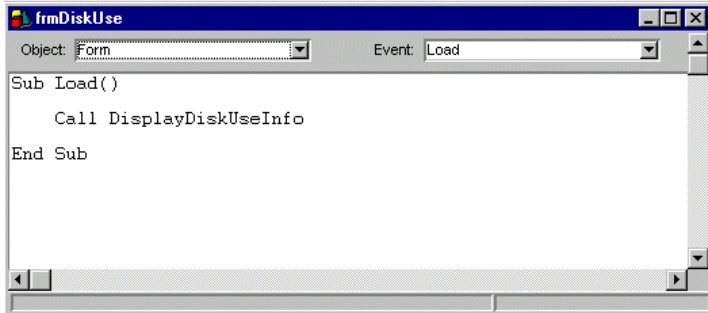
1. Open the form frmDiskUse, either by double-clicking its entry on the Form tab or by selecting it and clicking Modify.
2. Select the Form tab on the Attributes window and click Edit Code. A code window will be displayed with the form's Sub Load event.



3. Between the "Sub Load()" and "End Sub" commands, type the following code in Sub Load:

```
Call DisplayDiskUseInfo
```

Your completed code should look like this:



```

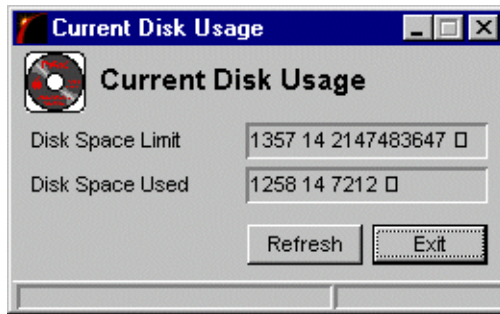
Object: Form      Event: Load

Sub Load()
    Call DisplayDiskUseInfo
End Sub

```

When you have finished typing in the code, press <Tab> and close the code window. Now that you have programmed Sub Load, the application is ready to run and test.

Click Run in the lower right hand corner of FirstClass RAD. If all steps have been completed correctly, the Disk Usage utility will appear with the raw, unformatted information returned from the batch administration requests you have made.



Your application is now retrieving information from your FirstClass Directory; however, it is probably not formatting the way you would like. The Batch Administrator returns more than just the value you have requested. It also returns the field ID, the field type, and a carriage return character. In the next activity, you will format this output to improve the display output.

When you are ready, close the Disk Usage utility by clicking Stop. FirstClass RAD automatically returns to development mode.

## Formatting output

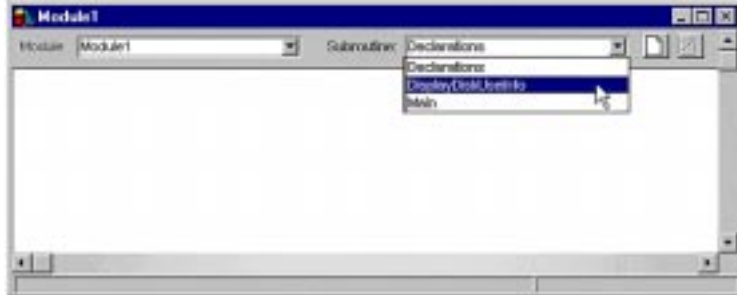
Let us revisit our subroutine and make a few changes. If you recall the output you were getting in the application's Disk Space Limit field, it was made up of three text strings separated by spaces:

```
1357 14 2147483647
```

These three values are the field ID (1357), the field type (14 = numeric) and the field value. What you are interested in is only the third number. You can extract this value with FirstClass RAD's StrToken internal function.

To format Output with StrToken:

1. Select the Module tab in FirstClass RAD.
2. Select Module1 and click Modify. The Module1 code window will open.
3. Select DisplayDiskUseInfo from the Subroutine selection list.



4. Edit the field assignment lines in your code adding StrToken to format the output. Change your current code to the following:

```
BatchAdmin("Get User " & fcUserID & " 1357")
frmDiskUse.txtLimit =
StrToken(fcBatchAdminReply, " ", 3)

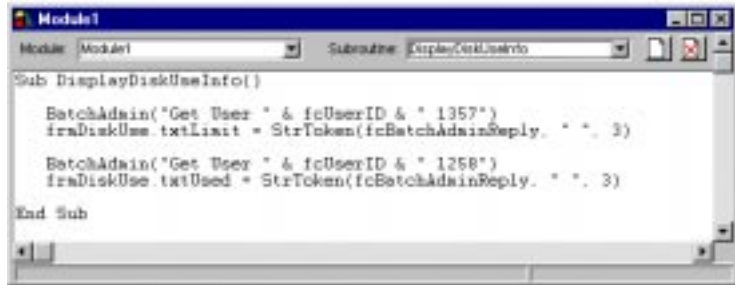
BatchAdmin("Get User " & fcUserID & " 1258")
frmDiskUse.txtUsed = StrToken(fcBatchAdminReply,
" ", 3)
```



As you can see in these changes, the StrToken function takes three arguments: the source text, the delimiting characters and the sub-string number divided by the delimiters.

In your code, StrToken takes the value of fcBatchAdminReply, separates it into sub-strings by space characters, and returns the third sub-string for the assignment.

Your finished code will look like this:



```

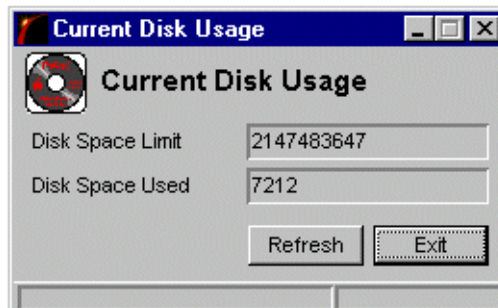
Sub DisplayDiskUseInfo()
    BatchAdmin("Get User " & fcUserID & " 1357")
    fnsDiskUse.txtLimit = StrToken(fcBatchAdminReply, " ", 3)

    BatchAdmin("Get User " & fcUserID & " 1258")
    fnsDiskUse.txtUsed = StrToken(fcBatchAdminReply, " ", 3)
End Sub

```

When you have finished typing in the code, press <Tab> and close the code window.

When you are ready to run the Disk Usage utility program, click Run in the lower right hand corner of FirstClass RAD. If all steps have been completed correctly, the Disk Usage utility will appear with the formatted information returned from Batch Administrator displayed.



The next step in your application is to program the Click events of buttons on the form.

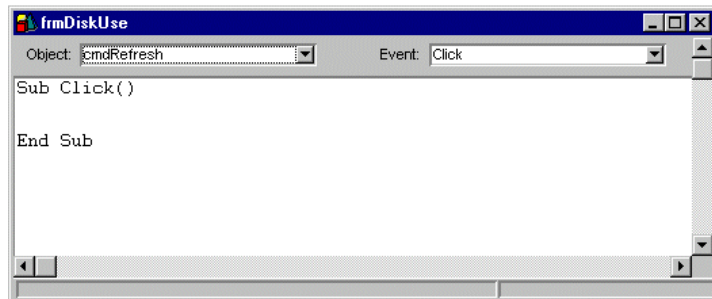
## Programming the Refresh button

Now that you have written a subroutine that retrieves and formats disk usage information, you will want to call that code from the Click event of the Refresh button. As you might have guessed, clicking the button triggers the Click event code to be run.

To program the Refresh button:

1. Open the form frmDiskUse, either by double-clicking its entry on the Form tab or by selecting it and clicking Modify.
2. Open the code window by double-clicking Refresh (cmdRefresh) on the form, or by selecting the button and clicking the Edit Code button on the Attributes form.

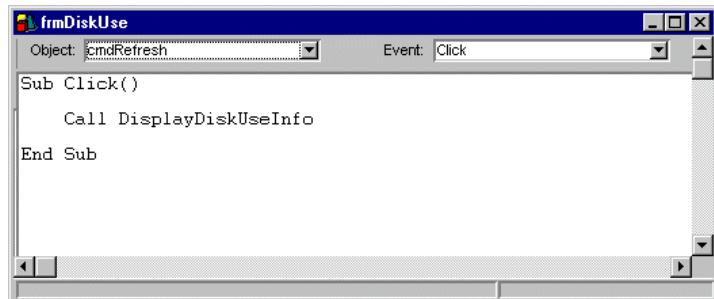
The code window will open and display the event code for the current field and default event.



The two selection lists on the top of the form. The Object selection list allows you to browse and select from the fields on the form, while the Event selection list allows you to browse the event code for each way the user can interact with the field. Any field on a form can have event code for any, all, or none of its user events.

3. Add our subroutine call to the Click Event.

Your completed code will look like this:



4. Press <Tab> and close the code window, when you have finished typing in the code,. Now that you have programmed the refresh button, the application is again ready to run and test.

## Running the application

Now you can run your application to see how it works.

To run your application:

1. Click the FirstClass RAD Run button to run your application in Debug Mode. Notice the disk usage amount displayed in the Disk Space Used field.
2. Create a new message in your mailbox and attach a small file to it.
3. Close the message.
4. Click the Refresh button.

Notice that the Disk Space Used amount will increase by the size of the new message and the file you have attached.

## Programming the Exit button

If the user clicks the Exit button, you simply want to close the form and have the application exit. You do this with the Unload command.

To unload the form:

1. Open the code window by double-clicking the Exit button (cmdExit) on the form, or by selecting the button and clicking the Edit Code button on the Attributes form.

The code window will open and display the event code for the current field and default event.

2. Add the following code to the Click event of cmdExit:

```
Unload frmDiskUse
```

Any code you add must go between the Sub Click and End Sub commands.

3. Press <Tab> and close the code window, after you have finished typing in the code.
4. Click Run on the FirstClass RAD interface and then click Exit, when you are ready to test the Exit button.

If you have done everything correctly, the Exit button will close the form.

## Formatting output (continued)

You may recall from the forms shown in this document that our sample user's disk space limit is "2147483647". This number may also be the number you are seeing if your account's disk space limit is set to "Unlimited." In this section you will make a final change to the DisplayDiskUseInfo subroutine to display "Unlimited" when the Batch Administrator returns the value "2147483647".

To write a simple 'If...Then' condition:

1. Select the Module tab in FirstClass RAD.
2. Select Module1 and click Modify.

The Module1 code window will open.

3. Select DisplayDiskUseInfo from the Subroutine selection list.
4. Edit the section of code that assigns the value to the txtLimit field.

This line should currently be:

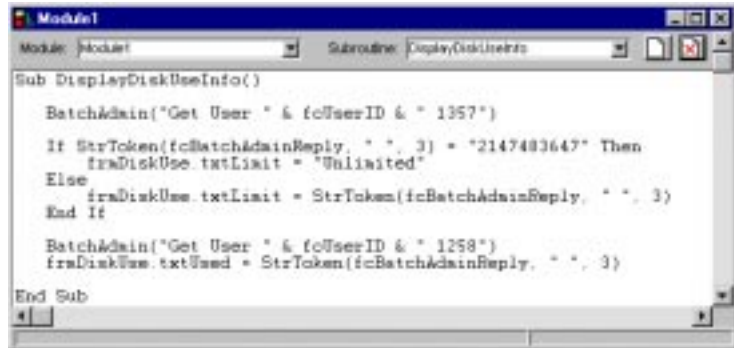
```
frmDiskUse.txtLimit =
StrToken(fcBatchAdminReply, " ", 3)
```

Replace this line of code with the following code:

```
If StrToken(fcBatchAdminReply, " ", 3) =
"2147483647" Then
    frmDiskUse.txtLimit = "Unlimited"
Else
    frmDiskUse.txtLimit =
StrToken(fcBatchAdminReply, " ", 3)
End If
```

In this code you are taking the value of the third sub-string of your Batch Administrator reply and comparing it to the value "2147483647" with an If...Then statement. If the condition evaluates to TRUE, indicating that the application user has the value 2147483647 as their disk space limit, assign the text "Unlimited" to the txtLimit field. Otherwise, the Else condition is executed which assigns the formatted value returned from the Batch Administrator to txtLimit.

Your finished code should look like this:



```
Module: Module1      Subroutine: DisplayDiskUseInfo
Sub DisplayDiskUseInfo()
    BatchAdmin("Get User " & fcUserID & " 1357")
    If StrToken(fcBatchAdminReply, " ", 3) = "2147483647" Then
        frmDiskUse.txtLimit = "Unlimited"
    Else
        frmDiskUse.txtLimit = StrToken(fcBatchAdminReply, " ", 3)
    End If
    BatchAdmin("Get User " & fcUserID & " 1258")
    frmDiskUse.txtUsed = StrToken(fcBatchAdminReply, " ", 3)
End Sub
```

When you have finished typing in the code, press <Tab> and close the code window.

When you are ready to test your application, click Run on the FirstClass RAD interface. When you are satisfied that the application is complete and working correctly, you can move on

to building and installing the application on your FirstClass system.

## Building and installing your application

After completing the project's programming and testing, you are ready to install the application on your FirstClass server.

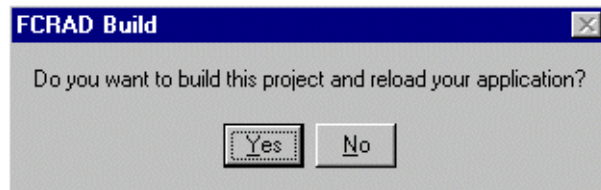
### Building your project

When a project is ready to be installed, the build process creates a compiled form of the project that the FirstClass server uses to host the application.

To build your project:

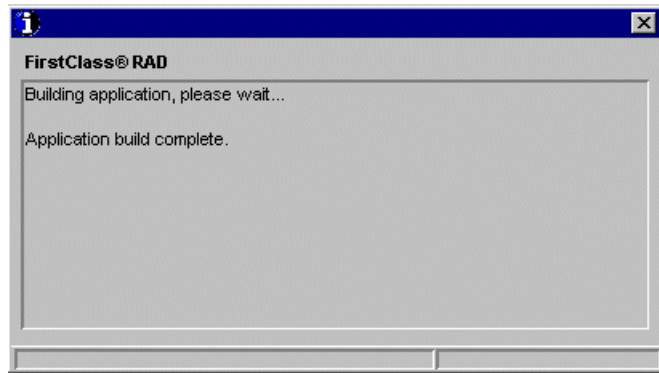
1. Click Build on the Project tab.
2. Click Yes to confirm the build.

A message box will appear and confirm that you are interested in building and reloading your application.



The build process creates an FCX, or FirstClass executable, file. Our project is named 'diskuse' so the resulting FCX file created is called diskuse.fcx. This file is written to the FirstClass RAD folder in your FirstClass post office. The FirstClass RAD folder contains built applications that are loaded by your FirstClass server at startup.

When the build process is complete you will see the following report window:



When the application build is complete, close this window. An application icon has been created in the 'Built Applications' folder in the FirstClass RAD folder. You may double-click this icon to launch your application from the FirstClass client.

## Installing additional application icons

9

This section describes the process of creating additional application icons.

To install icons:

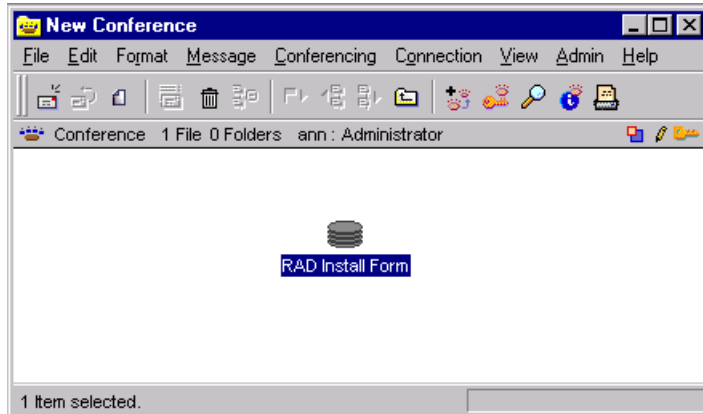
1. Create or open a conference that will contain the application icon.
2. Using the New Stationery menu under the Admin menu, select 'RAD Install Form'. The FirstClass RAD Install Form will open and allow you to input information.
3. Fill in the following required fields on the FirstClass RAD Install Form:
  - RAD.DLL  
This field contains the name of the FirstClass application server Dynamic Link Library. This field must contain the value 'RAD.DLL' for all applications.
  - Application name  
This field contains the name of the built application. The

application name is the same name as the project from which it was created. In this tutorial you created your project with the name 'diskuse'; correspondingly, the application name entered in this field must be 'diskuse'.



- Close the form and confirm the changes when prompted.

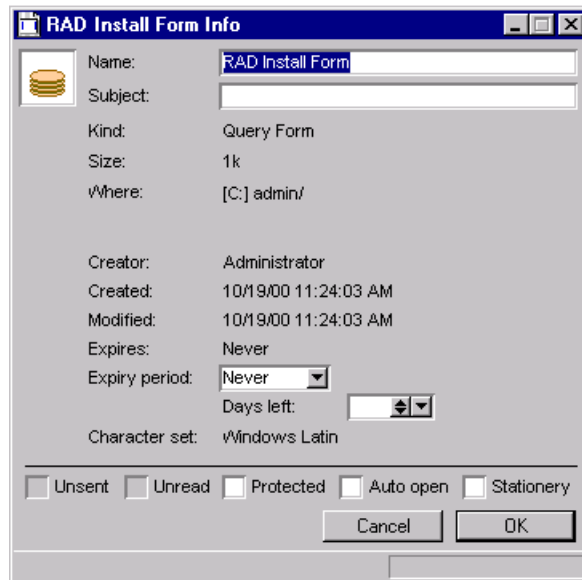
A new icon, RAD Install Form, has been created in your conference. Select the icon and edit its properties by selecting Properties from the File menu. The icon's Properties window will be displayed.



4. Rename the icon 'Disk Usage Utility' and assign an icon of your choice. Check the Protected checkbox at the bottom of



the form to enable the application. When finished, your form should look like this:



5. Close the Properties window and save the changes when prompted.
6. Double-clicking the icon will now launch the application. Any number of application icons can be created by repeating steps 2-5.

## Extra practice

Now that you have completed the Disk Usage utility, here are a few other ideas you can try for an extra challenge.

- Personalize the application by changing the title text on the form from “Current Disk Usage” to “<fcUserName>” Disk Usage” where <fcUserName> is the full user name of the application user.
- Make the value of the Disk Space Used field bold, if the user is within 100K of their disk limit. To get started, check the documentation on setting field attributes from code in the Programmer’s Guide.

## Extra practice

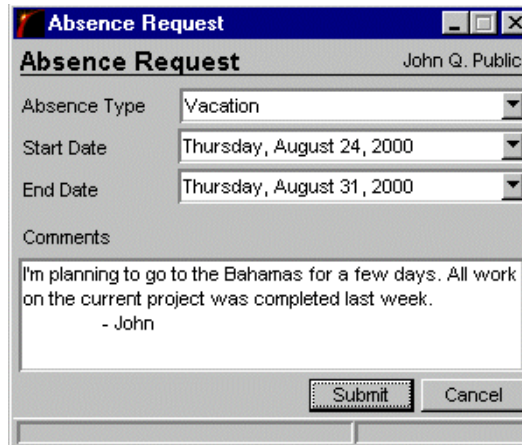
- Write an application that runs at client login and sends an email to the user when they are within 100K of their disk limit. Refer to the Programmer's Guide for help on sending email and writing a client login application.

## Tutorial 2: User Absence form with bound columns (database)

The Absence Request application is a bound column database program designed for use with a Microsoft Access database. In this type of application, associations are made between the fields on a form and the columns in a database. Column binding is done with FirstClass RAD's graphical interface and is the most efficient method of database programming in FirstClass.

The user running the application will be presented with an interface that they may use to submit an absence request. When the form is complete, the user can click Submit to insert a new record in the Absence Request database, or click Cancel to exit the application without submitting the request. The completed application will allow the user to specify the type of absence, the dates requested, and provide comments as needed.

10

The image shows a screenshot of a Windows-style application window titled "Absence Request". The window has a blue title bar with standard minimize, maximize, and close buttons. Below the title bar, the text "Absence Request" is displayed in a bold font, followed by the user name "John Q. Public" on the right. The main area of the window contains three dropdown menus: "Absence Type" with "Vacation" selected, "Start Date" with "Thursday, August 24, 2000" selected, and "End Date" with "Thursday, August 31, 2000" selected. Below these is a text area labeled "Comments" containing the text: "I'm planning to go to the Bahamas for a few days. All work on the current project was completed last week. - John". At the bottom right of the window are two buttons: "Submit" and "Cancel".

This tutorial is designed to guide you through every step of creating a simple bound column database application. As you go

through the tutorial, make sure that you complete every section of the application before moving onto the next section.

## Tutorial materials

The Absence Request application uses a FirstClass Designer form. In the first section of the tutorial you will create a new application form and cover the basics of using FirstClass Designer with FirstClass RAD.

Additionally, this application requires you to build a simple database in Microsoft Access. A working understanding of Microsoft Access or similar database product is a requirement of this tutorial.

## Assembling the project

This section outlines how to create a FirstClass RAD interface using FirstClass Designer. While experience with FirstClass Designer is not required to complete this tutorial, it is recommended.

### Adding a new form to your settings document

To add a new form to your settings document, locate your FirstClass RAD settings document in /FCPO/FCRAD/Rez/RAD.fc and follow these steps:

1. Open your RAD.fc settings document with FirstClass Designer.
2. Choose Form > New > Database Hit List.  
The newly created form opens.
3. Click the 'Form' menu again and select 'Split Bar'.  
This action will remove the split bar from the form.
4. Click the close box on the form and choose the 'Save' option.  
You will be prompted to enter the form's properties.
5. Set the following properties for the form:
  - Type: 'Form Template'

- Name: 'Absence Request Tutorial Form'
- Title: 'Absence Request'
- Range: 'Database Hit List'
- ID: Select a unique form ID in the numeric range between 10000-10999
- Style: 'Non-resizable'
- Menu ID: 'No Menu'.

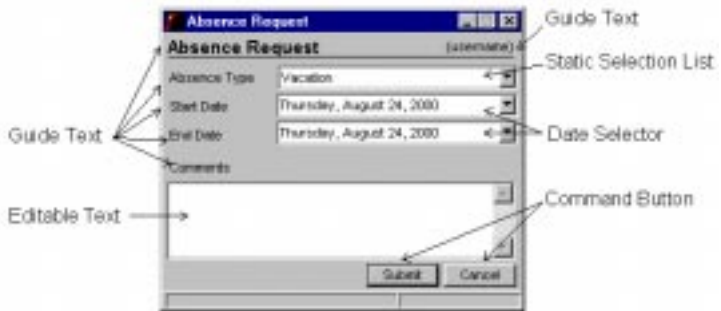
6. Click OK on the Properties window to save your changes.

### Adding fields to your form

To add fields to your new form:

1. Open your new form by double-clicking the form name in the FirstClass Designer resource list.
2. Select the Fields menu and choose the desired field type from the list of field categories.
3. Select the desired location for the field on your form, with the mouse.
4. Adjust the size and location of the field using the mouse and keyboard. Repeat steps 2-4 and create the following application interface:

10



### Setting field attributes

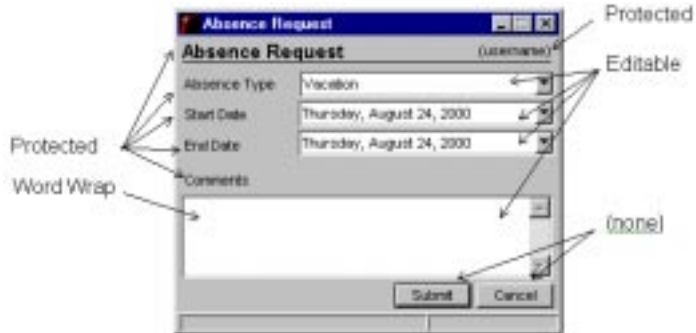
Use the Attributes window to manipulate each field's attributes. Attributes include style, position and default elements such as Hidden, Protected, Bold, and Color. For details on FirstClass

Designer field attributes, see "FirstClass Designer field attributes" on page 129 in the Appendix.

If the Attributes window is not visible in FirstClass Designer, you can open it by selecting the 'Fields' menu and choosing 'Field Attributes'.

### Assigning attributes

Use the Attributes window to set the following attributes for each field.



### Setting the Absence type list

The next step is to use the Attributes window to assign a default list for the Absence Type.

To set the Absence type list:

1. Select the static selection list immediately to the right of the guide text Absence Type.
2. Select the 'Contents' tab on the Attributes window.
3. Type the following code into the 'List' attribute to establish selection list defaults for the field:

```
Vacation=0;Illness=1;Funeral=2;Maternity  
Leave=3;Personal Holiday=4;
```

4. Press <Tab> to update your changes to the selection list.

The default absence type, Vacation, should display in your list and the other three options entered should be available as list options.

Once the form is complete, close your form, save your changes and exit FirstClass Designer. You are now ready to log in to your server and add the form to a FirstClass RAD project.

## Creating a new project

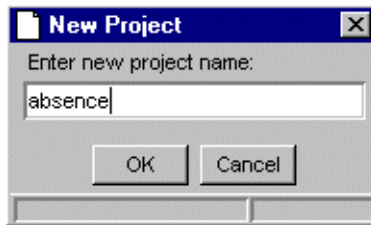
To create a new project in FirstClass RAD, login to your FirstClass server as the Administrator and run FirstClass RAD by double-clicking on its icon. The FirstClass RAD icon is located in the folder:

Application Server\Applications\FirstClass RAD

To create a new project:

1. Click New on the Project tab of the FirstClass RAD interface.
2. Type in a new project name absence.

Project names may be up to 23 characters long and may not contain spaces.



3. Click OK.

The new project has been created. FirstClass RAD is now ready for you to begin programming your application.

## Working with forms and fields

The first step in preparing your new project is to add forms and name the fields on the forms.

### Adding a new form to your project

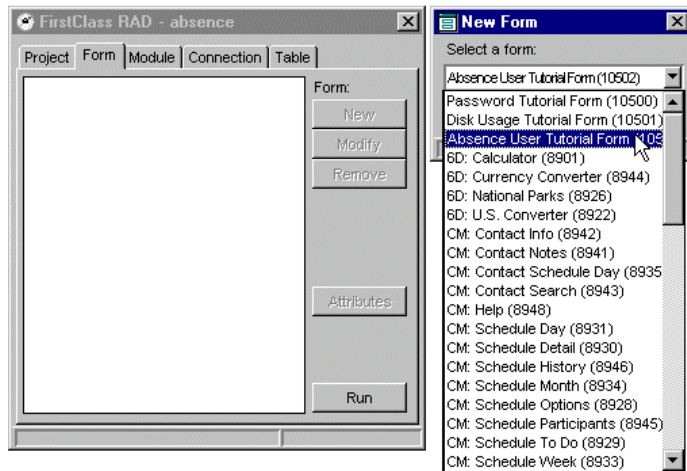
Adding the Absence Request Tutorial Form opens the form in FirstClass RAD. When the form is opened, FirstClass RAD downloads all of the form information into the project database.

Once a form has been added, all of its fields may be named and programmed.

To add a new form:

1. Click New on the Forms tab.
2. Select the Absence Request Tutorial Form from the list of forms in the settings file.

If you do not see the form listed, make sure that you are logged into the FirstClass server with the correct settings document.



3. Click OK to add the form.

The form opens and the form's information is added to the project database. The FirstClass RAD Attributes window will open to allow form and field naming and programming.

The next step is to name your form and the fields on the form.

### **Naming forms and fields**

To name your form:

1. Click on the Form tab of the Attributes window to display the default name of the form, the form title, and the form's bound data environment.
2. Edit the form's default name and change it to 'frmAbsence'.

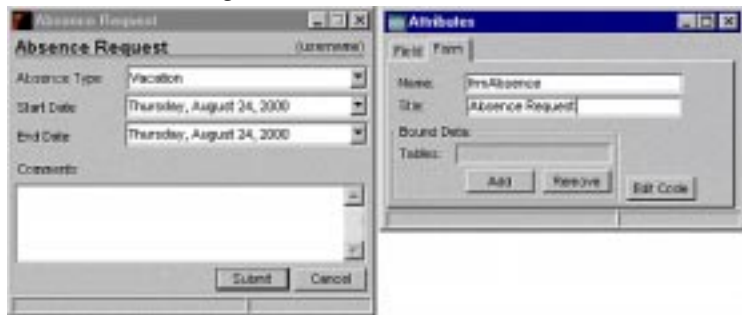


The identifier frmAbsence is what you will use to reference the form when you are writing code.

It is important to note that in FirstClass RAD you commonly assign a prefix to indicate an object's type. In this case, the prefix 'frm' indicates that this is a form object.

3. When you have finished entering the form name, press <Tab> to update the change.
4. Enter the form's title as 'Absence Request'.

The value of the title field in FirstClass RAD will be assigned to the title bar on the form when the application is run. This value overrides any values you have previously set in FirstClass Designer.



10

5. When you have finished entering the form title, press <Tab> to update the change.

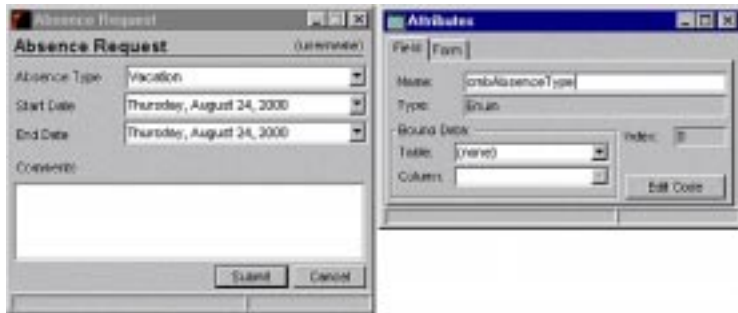
Without closing the form or the Attributes window, move onto the next step. If you have accidentally closed the form, simply reopen the form by selecting it from the Form tab and click Modify. If you have accidentally closed the Attributes window, you may reopen this by clicking the Attributes button on the Form tab.

It is important to name the fields on the form that you intend to reference in your code. FirstClass RAD assigns a default name for each field if you do not assign one; however, these names are generally not specific enough to make your code easy to read. When assigning field names it is important to assign unique values that are easy to remember. In this application, you will only program text fields, buttons and a selection list.

To name fields:

1. Select the Field tab of the Attributes window.
2. Click on the selection list field immediately right of the guide text Absence Type. The Attributes window will display the default name of the field.
3. Change the default name of the field from its current name (for example, SelectionList1007) to 'cmbAbsenceType' by modifying its name in the Attributes window.

Again, you assign a prefix to indicate an object's type. In this case, the prefix 'cmb' indicates that this is a selection list or 'combo box' field object.



4. When you have finished entering the field's name, press <Tab> to update the change.

Complete steps 2-4 of the field naming process and name the form's fields according to the following diagram. Don't forget to press <Tab> after assigning the name of each field. When you have named all of the fields on the diagram, close the form by clicking the form's close box.



## Creating a Microsoft Access database

It is not within the scope of this tutorial to cover Microsoft Access database design and programming in-depth. Please refer to Microsoft's documentation and tutorials to familiarize yourself with this application if you plan to use it in developing applications.

**Note** FirstClass RAD does not require Access as the application database. We use it just as an example, because of the popularity of Access among FirstClass RAD developers. For names of other usable databases, see "ODBC drivers" on page 7.

For your Absence form you will create a simple one-table database in Access.

### Building your database table

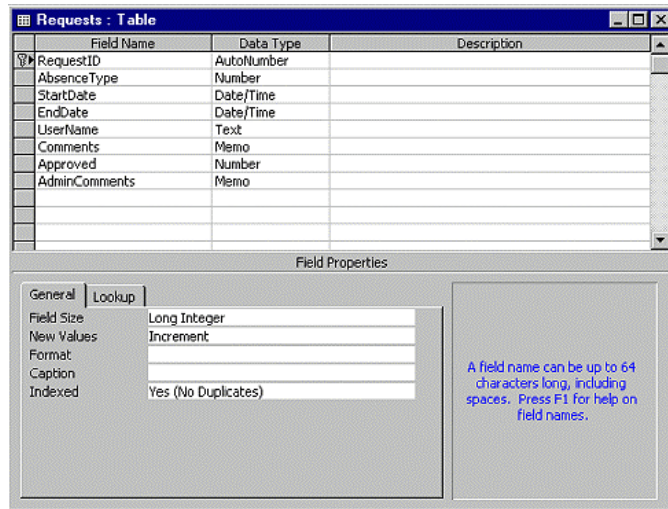
There are many ways of creating a table in Microsoft Access. The following procedure describes one method.

To build your database table:

1. Run Microsoft Access.
2. Create a new blank database in /FCPO/FCRAD/DATA folder.
3. Name the new database file Absence.mdb.
4. Click New on the Tables tab. When prompted, select Design View.
5. Create the following table.

# 10

Default values and field sizes automatically assigned by Microsoft Access will be adequate for this application.



- Close and save the table. When prompted, name the table 'Requests'.

**Note** If you are prompted to set a Primary Key, let Microsoft Access set it automatically for you.

With your database created and table built, you can move onto configuring the Open Database Connectivity (ODBC).

## Configuring ODBC

When the FirstClass RAD installer was run it installed Open Database Connectivity (ODBC). ODBC is the Windows industry standard for database connectivity.

### Using the ODBC Driver Manager

You will use ODBC and the Microsoft Access ODBC Driver to configure a data source for use with your application.

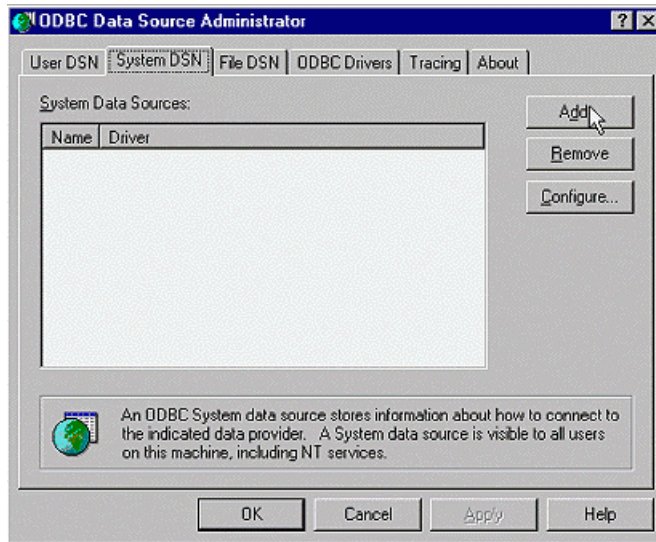
Before FirstClass RAD can connect to your database, a data source must be configured in ODBC.

To create an ODBC data source:

1. Open the Windows NT Control Panel. Double-click the ODBC icon.

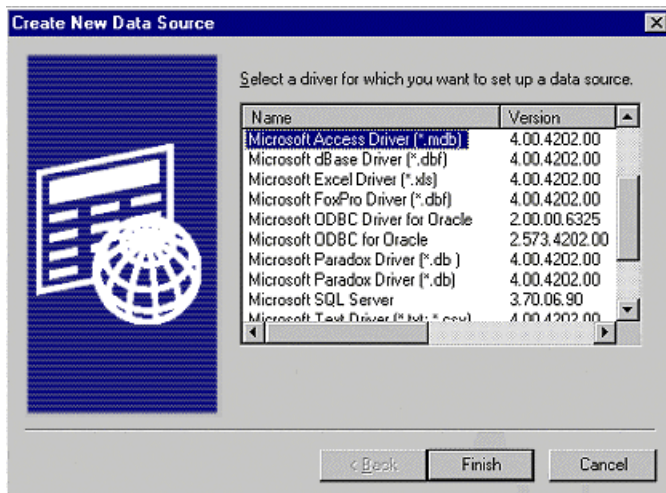
Depending on your installation, ODBC may be named 'ODBC', 'ODBC Data Sources' or 'ODBC Data Source Administrator'.

2. Select the System DSN tab on the ODBC Data Source Administrator window and click Add.

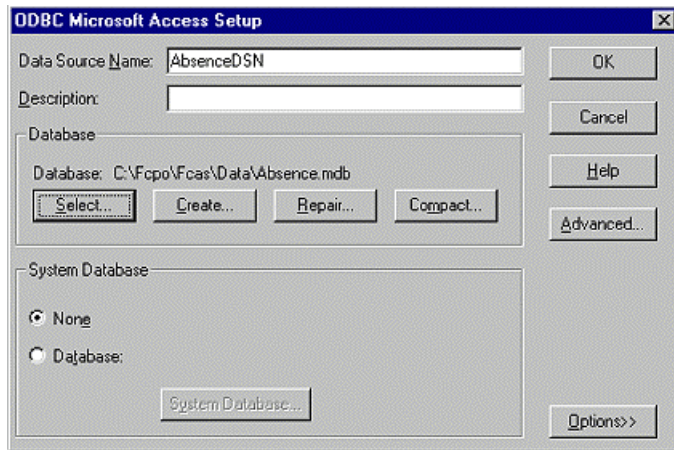


10

3. Select Microsoft Access Driver (\*.mdb)' and click Finish.



The Microsoft Access Driver dialog box appears.



4. Enter 'AbsenceDSN' in the Data Source Name field at the top of the form.
5. Click Select in the middle left area of the form. Use the file dialog to find the database Absence.mdb, which you created in /FCPO/RAD/DATA.
6. Click OK to finish installing the data source.

Congratulations, you have created an ODBC data source. This data source will be used to allow the Absence Request application to connect to the Absence.mdb database.

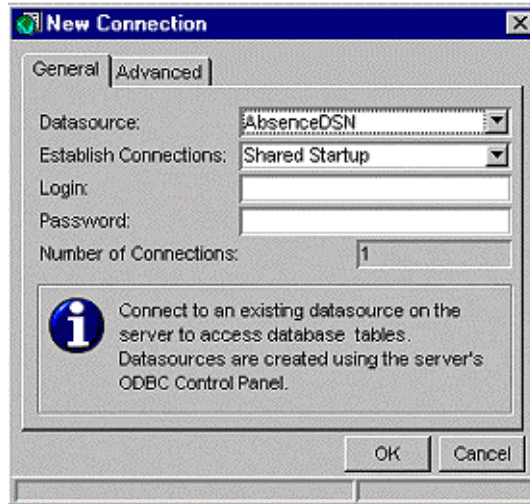
## Connecting a project to a database

Once you have configured the AbsenceDSN data source for the FirstClass server, you must configure FirstClass RAD to connect to the database. Connections are managed from the Connections tab.

To create a connection to the Absence database:

1. Click New on the Connections tab to display the New Connection form.
2. Select the AbsenceDSN data source from the Datasource selection list.

- Specify 'Shared Startup' in the Establish Connections list.  
The number of connections will automatically be set to 1.



No login or password is necessary for this database. You can use the default options on the Advanced tab; they are appropriate for the Microsoft Access database driver.

10

- Click OK to save the AbsenceDSN connection.

The AbsenceDSN connection is added to the list of connections displayed in the window on the Connections tab of FirstClass RAD.

### Adding the Requests Table to your project

Once the AbsenceDSN connection has been configured, the Requests table in the database may be added to the project using the Tables tab.

To add the new table:

- Click New on the Table tab to display the Table Configuration form.
- Select the AbsenceDSN connection from the selection list at the top of the form.

Tables available for that data source will appear in the Table selection list.

3. Select the Requests table from the Tables selection list.



There is no need to modify any fields on the Advanced tab, they are appropriate for the Microsoft Access database driver.

4. Click OK to save the Requests table configuration.

The table is now added to the list of tables displayed in the drop down list on the Tables tab.

## Binding fields to columns

FirstClass RAD includes functionality for directly associating, or “binding”, database columns with fields on your form. Bound fields in FirstClass RAD allow the developer to create applications that update, add, and delete data from the bound table.

### Adding your table to the data environment

Binding database columns to fields is a two-step process. First, the table must be added to the data environment of the form. Second, each field on your form must be bound to a specific database column. A single form can have any number of tables in its data environment and any number of bound fields.

To add your table to the data environment:

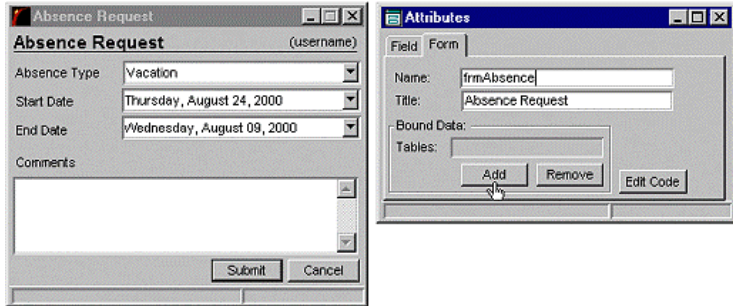
1. Open the form frmAbsence, either by double-clicking its entry on the Form tab or by selecting it and clicking Modify.



- Click on the Form tab of the Attributes window.

The form displays form name and data environment information.

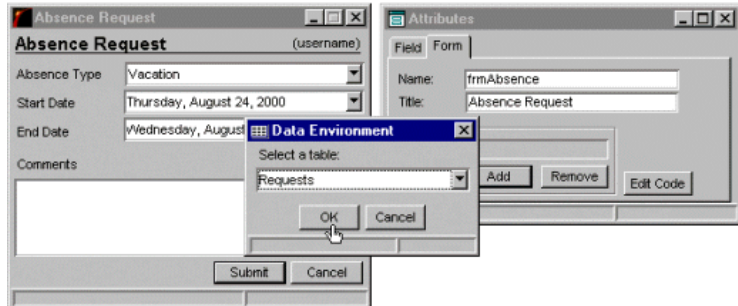
- Click Add, in the Bound Data section of the Attributes window.



The Data Environment dialog will be displayed and you will have the ability to associate the Table object you created in the last section with this form.

- Select the Requests table from the drop down list in the Data Environment dialog.
- Click OK.

10



The Requests table has been added to the data environment of the form and is displayed in the Attributes window. When the Absence Request form is loaded, a reference to the Requests table data will be created and associated with the form.

The next step is to bind each field on your form to a database column.

## Binding form fields

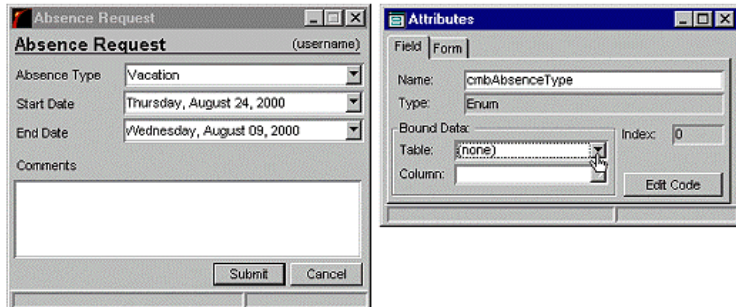
To bind form fields:

1. Select the Absence Type selection list field on the form (cmbAbsenceType).

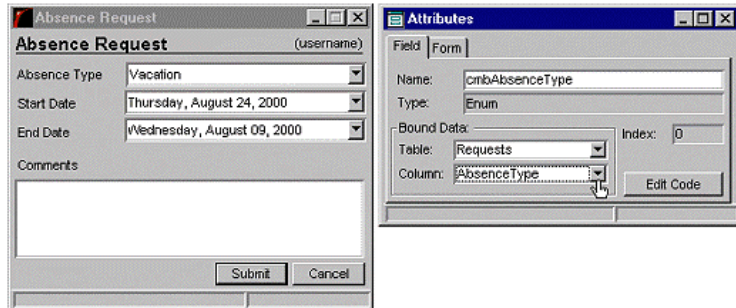
The Field tab of the Attributes window will display the field's attributes.

2. Select the Requests table with the Table selection list on the Field tab.

The columns in the table are immediately displayed in the Column selection list.



3. Select the AbsenceType column from the Column selection list.



The cmbAbsenceType field is now bound to the AbsenceType database column, and data from the database column will be linked with the field when the project is run.

Repeat this process to bind the following fields to the appropriate database columns:

Field name	Bound column
cmbAbsenceType	AbsenceType
txtUserName	UserName
datStart	StartDate
datEnd	EndDate
txtComments	Comments

## Programming your project

In this project, the program begins by showing the Absence Request form. All applications begin running in the Sub Main subroutine in Module1. When your application is run, your startup form is shown from this code.

### Showing the form with Sub Main()

To open the form at application startup time, you add one line of BASIC code to the Sub Main subroutine.

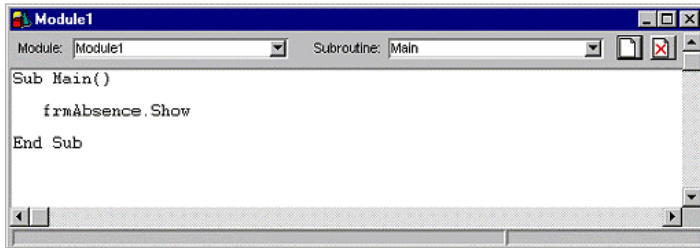
To show the form:

1. Select the Module tab in FirstClass RAD.
2. Select Module1 and click Modify. The Module1 code window will open.
3. Select 'Main' from the subroutine selection list.
4. Between “Sub Main()” and “End Sub”, type the following code in Sub Main:

```
frmAbsence.Show
```

**Note** Always press <Tab> and close the Module window to save your code changes. If you do not, code changes will be lost when you run your application.

Your completed Sub Main should look like this:



Now that you have programmed Sub Main, the application is ready to run and test.

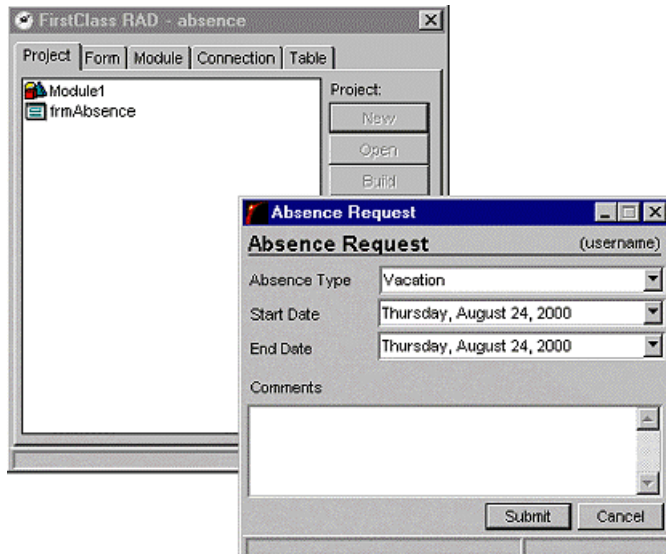
### Running the Absence Request Application in FirstClass RAD

At this point, running the Absence Request application simply opens the form interface. Once other fields on the form have been programmed, the interface will respond intelligently to user interaction. For now, let us show the form in debug mode by running the application in FirstClass RAD.

To run the Absence Request application:

1. Click Run, in the lower right hand corner of FirstClass RAD.

If all steps have been completed correctly, the Absence Request interface will appear. The Run button caption changes to 'Stop'.



Your application is now being run in Debug Mode. Debug Mode applications run from within the FirstClass RAD environment just as if they were built and deployed on your system. Debug Mode allows you to test the application during development without fully installing the application on your server.

When you are finished testing your application, click Stop to stop the running program and put FirstClass RAD back into development mode.

## Creating a new record

The next step is to take the bound form that you have created and use it to insert a record into the database. Inserting a record with bound tables in FirstClass RAD is a two-step process.

The first step is to issue the Insert command against the table. The Insert command creates a temporary new blank database record in memory. The record is, of course, already bound to the fields on our form, so any default values on the form will automatically be cleared to make way for the new record. Once the Insert command has been issued, the user can type in and select values for the new record.

After the user has entered the Absence Request record, the next step is to write the new temporary record into the database. This is done with the Update command.

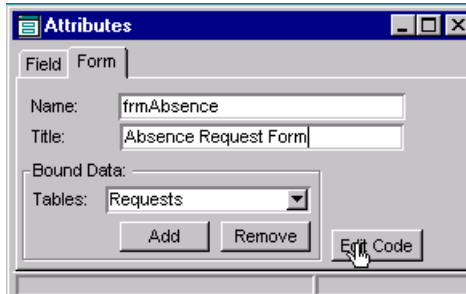
### Inserting a new temporary record

When the application is run, the first thing you want to do is prepare the form for user input of the new record. To do this you will program the Load Event of frmAbsence.

To program the Load Event:

1. Open the form frmAbsence, either by double-clicking its entry on the Form tab or by selecting it and clicking the Modify button.

2. Select the Form tab on the Attributes window and click the Edit Code Button. A code window will be displayed with the form's Sub Load event.

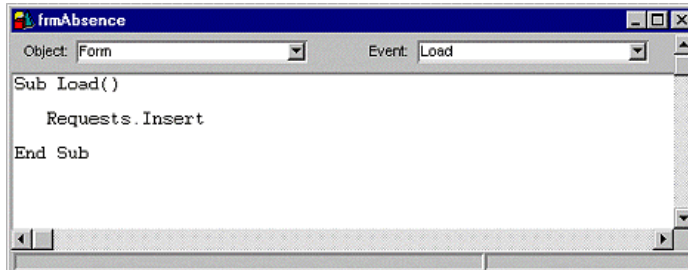


3. Between the “Sub Load()” and “End Sub” commands, type the following code in Sub Load:

```
Requests.Insert
```

In this code, you are creating a temporary record in the Requests table. The Requests table is available for us to reference in code because you added it to the form's data environment when you did the column binding.

Your completed code should look like this:



When you have finished typing in the code, press <Tab> and close the code window. Now whenever the form is loaded, a temporary inserted record will be created and the fields on the form will be reset to the database's default values.

## Confirming an inserted record with the Update

## command

Now that a temporary record has been created, your user will be able to enter data to describe their absence request. Once they have completed their input, the next step will be to confirm the insert and write the new record to the database. You must also include code to close the application when the absence request is submitted.

### Programming the Submit button

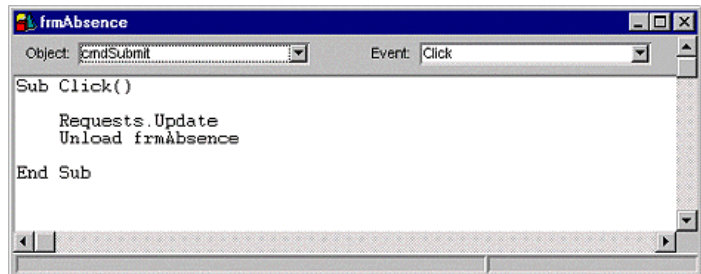
In this section you will code the Click event on the Submit button to insert the temporary record into the database.

To program the Submit button:

1. Open the form frmAbsence, either by double-clicking its entry on the Form tab or by selecting it and clicking Modify.
2. Open the code window by double-clicking the Submit button (cmdSubmit) on the form, or by selecting the button and clicking Edit Code on the Attributes form.
3. Confirm that the Object selection list is set to cmdSubmit and that the Event selection list is set to Click.
4. Add the following code to the Click Event:

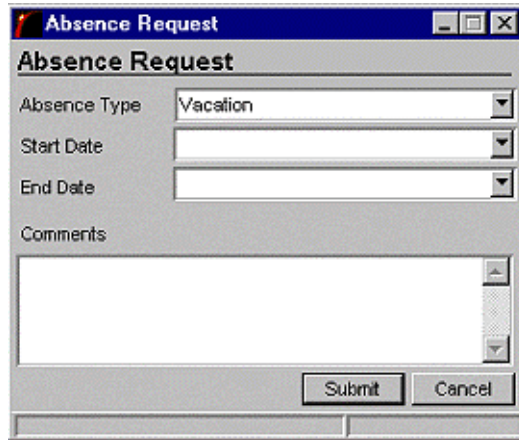
```
Requests.Update
Unload frmAbsence
```

Your completed code should look like this:



When you have finished typing in the code, press <Tab> and close the code window. The application is again ready to run and test.

5. Click Run to run your application in Debug Mode. The fields on the form will be reset to their default values, as seen below:



The screenshot shows a Windows-style dialog box titled "Absence Request". The dialog has a title bar with standard minimize, maximize, and close buttons. The main area contains the following fields:

- "Absence Type": A dropdown menu with "Vacation" selected.
- "Start Date": An empty date dropdown menu.
- "End Date": An empty date dropdown menu.
- "Comments": A large, empty text area with a vertical scrollbar on the right.

At the bottom right of the dialog, there are two buttons: "Submit" and "Cancel".

To test your application:

1. Enter data for a new Absence Request record.
2. Click Submit.

The form will close and FirstClass RAD will return to development mode.

3. Open your database, Absence.mdb, in Microsoft Access and confirm that your new record has been written to the Requests table.

## Programming the Cancel button

If the user clicks the Cancel button, you simply want to close the form and have the application exit. At this point in your application, your user may have entered data into the fields on the form, and consequently, the columns of the temporary record. This is not a problem; when the application closes, the temporary record will be automatically discarded.

To unload the form:



1. Open the code window by double-clicking the Cancel button (`cmdCancel`) on the form, or by selecting the button and clicking Edit Code on the Attributes form.

The code window will open and display the event code for the current field and default event.

2. Add the following code to the Click event of `cmdCancel`:

```
Unload frmAbsence
```

3. When you finish typing in the code, press <Tab> and close the code window.

When you are ready, click Run and test the Cancel button. If you have done everything correctly, clicking Cancel will close the form.

## Assigning the 'txtUserName' field

You will notice that when you run the application, the field `txtUserName` is blank. This field is protected and does not allow the user to enter data, so consequently, the column that it is bound to in the database is blank when the record is inserted.

In this section you will set the value of `txtUserName` automatically to reflect the name of the user running the application. When you click Submit, the FirstClass user name of the user making the request will be written to the requests table, along with the other request data.

To assign the `txtUserName` field in Sub Load:

1. Open the form `frmAbsence`, either by double-clicking its entry on the Form tab or by selecting it and clicking Modify.
2. Select the Form tab on the Attributes window and click Edit Code.

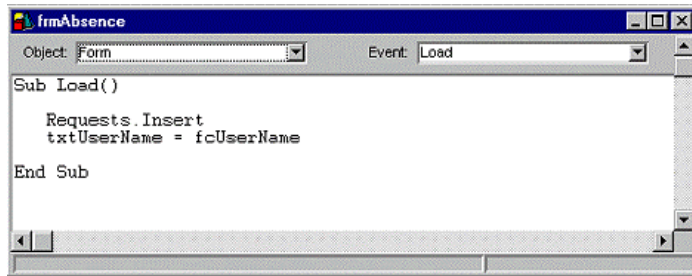
A code window will be displayed with the form's Sub Load event.

3. Type the following code in Sub Load immediately after `Requests.Insert`:

```
txtUserName = fcUserName
```

Here you use the `fcUserName` internal function to get the user name of the user currently using the application. Other fields on the interface could be similarly assigned default values as needed. For more information on internal functions, see our online help.

Your completed code should look like this:



When you finish typing in the code, press <Tab> and close the code window. Run and test your application.

## Building and installing your application

After completing a project's programming and testing, you are ready to install the application on your FirstClass server.

### Building your project

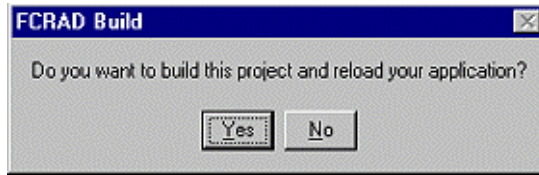
When a project is ready to be installed, the build process creates a compiled form of the project that the FirstClass server uses to host the application.

To build your project:

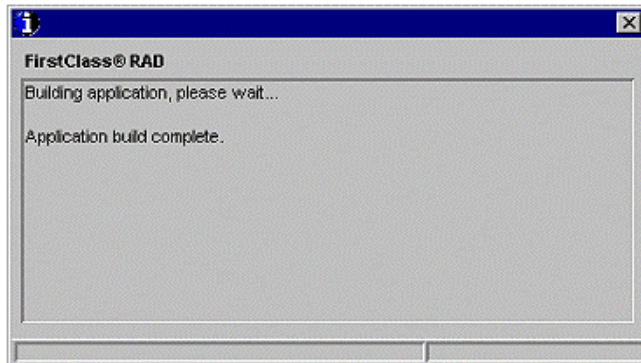
1. Click Build on the Project tab.

A message box will appear and confirm that you are interested in building and reloading your application.

- Click 'Yes' to confirm the build.



The build process creates an FCX, or FirstClass executable, file. Our project is named 'absence' so the resulting FCX file created is called absence.fcx. This file is written to the FirstClass RAD folder in your FirstClass post office. The RAD folder contains built applications that are loaded by your FirstClass server at startup. When the build process is complete you should see the following report window:



**10**

- Close this window and proceed to the next step, when the application build is complete,

An application icon has now been created in the 'Built Applications' folder in the FirstClass RAD folder. You may double-click this icon to launch your application from the FirstClass client. If you want to install additional application icons, follow the steps in "Installing additional application icons" on page 85.

## Extra practice

Now that you have completed the Absence Request application, here are a few other ideas you can try for an extra challenge.

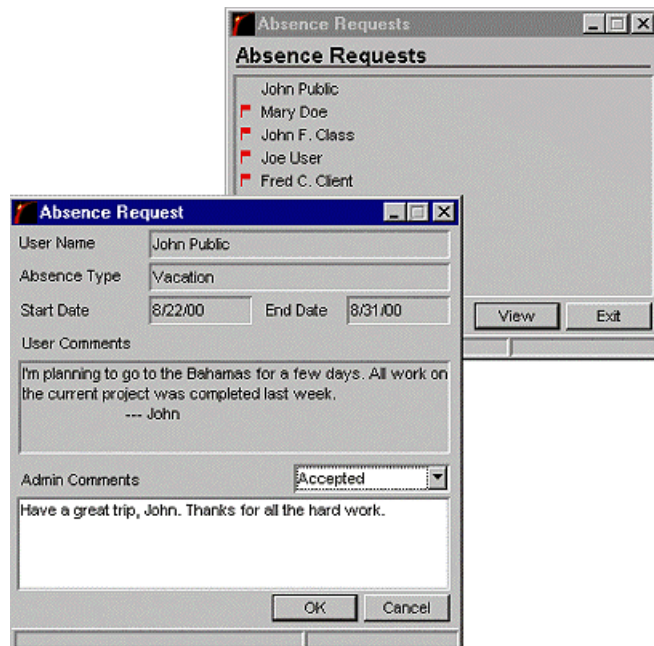
- Use the Today command to preset the date selector controls on your form. Try to assign the Start date as tomorrow and the End date as the day after tomorrow by default. (Hint: Today + 1 is tomorrow, Today + 2 is the day after tomorrow.)
- Write an If...Then condition to make sure that the Start Date entered is before the End Date. Display a message box warning the user if this does not occur.
- Send an email message to a conference or supervising user whenever a request is made. See the Programmer's Guide for techniques on sending an email.

## Tutorial 3: Administrator Absence form with non-bound columns (database)

The Administrator Absence form expands the concepts presented in the Absence Request application by introducing fixed list controls, multiple forms, and integrating bound and non-bound forms. In this application, an administrator can approve or reject the absence requests entered by the Absence Request application.

When the administrator approves or rejects a request, the approval status and comments are written to the database.

11



This tutorial is designed to guide you through every step of creating a database application. As you go through the tutorial, make sure that you complete every section of the application before moving onto the next section.

## Tutorial materials

The Absence Admin application uses several FirstClass Designer forms. In the first section of the tutorial you will create the new application forms and cover more of the basics of using FirstClass Designer with FirstClass RAD.

Also, it is mandatory that you completed Chapter 10, "Tutorial 2: User Absence form with bound columns (database)" before continuing with this one, as you will be using the same database.

## Assembling the project

This section outlines the creation of a FirstClass RAD interface using FirstClass Designer. While experience with FirstClass Designer is not required to complete this tutorial, it is recommended.

### Adding a new form to your settings document

To add a new form to your settings document, locate your FirstClass settings document:

```
/FCPO/FCRAD/Rez/RAD.fc
```

and follow these steps:

1. Open your RAD.fc settings document with FirstClass Designer.
2. Choose Form > New > Database Hit List.
3. The newly created form will open. Click on the 'Form' menu again and select 'Split Bar'; this action will remove the split bar from the form.
4. Click the close box on the form and choose 'Save'.

You will be prompted to enter the form's properties.

5. Set the following properties for the form:
  - Type: 'Form Template'
  - Name: 'Absence List Form'
  - Title: 'Absence Requests'
  - Range: 'Database Hit List'
  - ID: Select a unique form ID in the numeric range between 10000-10999
  - Style: 'Non-resizable'
  - Menu ID: 'No Menu'.
6. Click OK on the Properties window to save your changes.

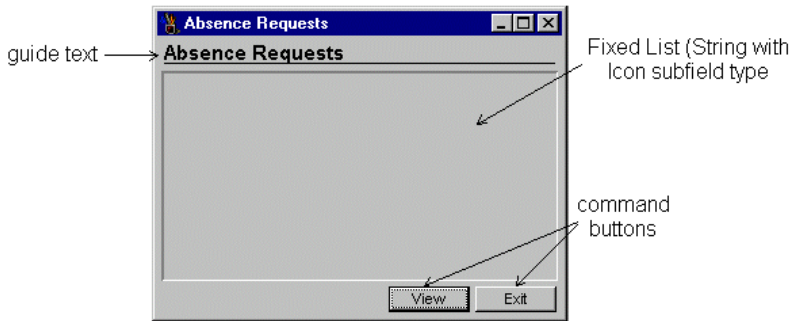
### **Adding fields to your form**

Now you will add fields to your new form.

To add fields to your form:

1. Open your new form by double-clicking the form name in the FirstClass Designer resource list.
2. Select the 'Fields' menu and select the desired field type from the list of field categories.
3. Select the desired location for the field on your form, with the mouse.
4. Adjust the size and location of the field using the mouse and keyboard.

Repeat steps 2-4 and create the following application interface:



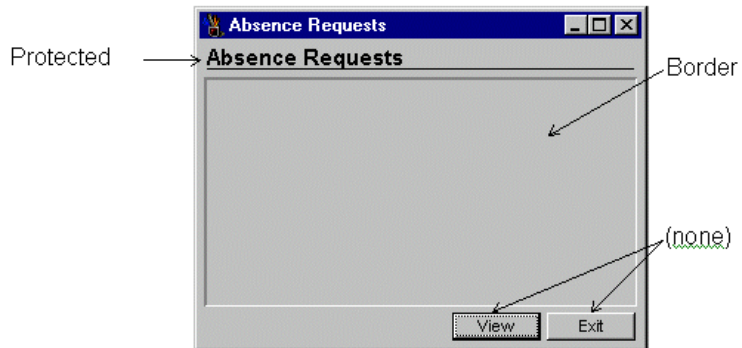
## Setting field attributes

Use the Attributes window to manipulate each field's attributes. Attributes include style, position and default elements such as Hidden, Protected, Bold, and Color. For details on FirstClass Designer field attributes, see "FirstClass Designer field attributes" on page 129 in the Appendix.

If the Attributes window is not visible in FirstClass Designer, you can open it by selecting the 'Fields' menu and choosing 'Field Attributes'.

### Assigning attributes

Using the Attributes window, set the following attributes for each field.





**Creating the Absence Request form in your settings document**

The next step is to create the Absence Request form in your settings document.

To create this form:

1. Choose Form > New > Database Hit List.

The newly created form will open.

2. Click on the 'Form' menu again and select 'Split Bar'

This action removes the split bar from the form.

3. Click the close box on the form and choose 'Save'. You will be prompted to enter the form's properties.

4. Set the following properties for the form:

- Type: 'Form Template'
- Name: 'Absence Detail Form'
- Title: 'Absence Request'
- Range: 'Database Hit List'
- ID: Select a unique form ID in the numeric range between 10000-10999
- Style: 'Non-resizable'
- Menu ID: 'No Menu'.

5. Click OK on the Properties window to save your changes.

### Adding fields to the Absence Detail form

Following the steps described in Adding Fields to the Absence List Form, create the following interface according to the diagram:



### Assigning field attributes

Using the Attributes window, set the following attributes for each field according to the diagram:



### Setting the Absence Type list

Use the Attributes window to assign a default list for the Absence Type selection list:

1. Select the static selection list immediately to the right of the guide text 'Absence Type'.
2. Select the 'Contents' tab on the Attributes window
3. Type the following code into the 'List' attribute to establish selection list defaults for the field:

```
Vacation=0;Illness=1;Funeral=2;Maternity  
Leave=3;Personal Holiday=4;
```

4. Press <Tab> to update your changes to the selection list.

The default absence type, 'Vacation', will display in your list and the other options entered will be available as list options.

### Setting the Absence Approval list

Using the steps described in Setting the Absence Type list, assign the default list for the absence approval selection list. The absence approval selection list is located to the right of the guide text 'Admin Comments'.

1. Enter the following list values:

```
Unconfirmed=0;Approved=1;Rejected=2;
```

2. Press <Tab> to update your changes to the selection list. The default absence type, 'Unconfirmed', will display in your list and the other options entered will be available as list options.

Once the forms have been completed, save your changes and exit FirstClass Designer. You are now ready to log in to your server and add the forms to a FirstClass RAD project.

11

## Creating a new project

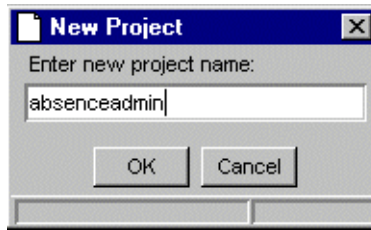
Log in to your FirstClass server as the Administrator (“Admin”) and run FirstClass RAD by double clicking on its icon. The FirstClass RAD icon is located in the folder:

```
Application Server\Applications\FirstClass RAD
```

To create a new project:

1. Click New on the Project tab of the FirstClass RAD interface.
2. Type in a new project name, 'absenceadmin'.

Project names may be up to 23 characters long and may not contain spaces.



3. Click OK to create the new project.

The new project has been created. FirstClass RAD is now ready for you to begin programming your application.

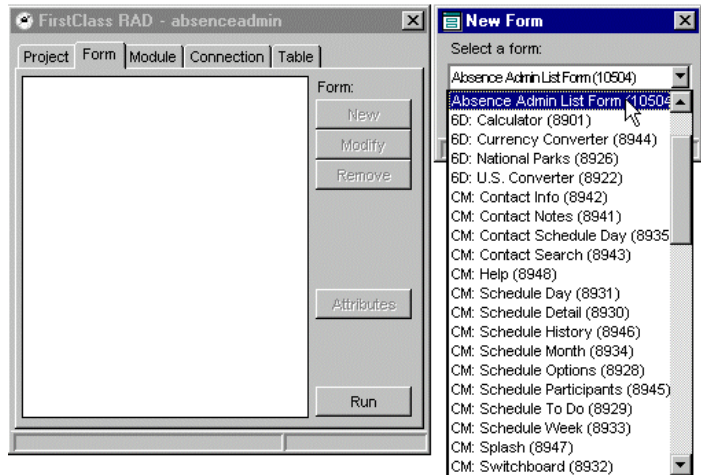
## **Adding the Absence List form to your project**

Adding the Absence List Form opens the form in FirstClass RAD. When the form is opened, FirstClass RAD downloads all of the form information into the project database. Once a form has been added, all of its fields may be named and programmed.

To add a form to your project:

1. Click New on the Forms tab.
2. Select the 'Absence List Form' from the list of forms in the settings document.

If you do not see the form in the list, make sure that you are logged into the FirstClass server with the correct settings document.



3. Click OK to add the form.

The form opens and the form's information is added to the project database. The FirstClass RAD Attributes window will open to allow field naming and programming in the next section.

11

**Naming forms and fields** The next step is to name your form and the fields on the form.

To name your form:

1. Click on the Form tab of the Attributes window to display the default name of the form, the form title, and the form's bound data environment.
2. Change the default form name to 'frmAbsenceList'.

The identifier frmAbsenceList is what you will use to reference the form when you are writing code.



3. Set the form's title to 'Absence Requests'.

The value of the Title field in FirstClass RAD will be assigned to the title bar on the form when the application is run.

4. Press <Tab> to update the changes.

To name fields:

1. Select the Field tab of the Attributes window.
2. Click on the fixed list field immediately below the guide text 'Absence Requests'.

The Attributes window will display the default name of the field.

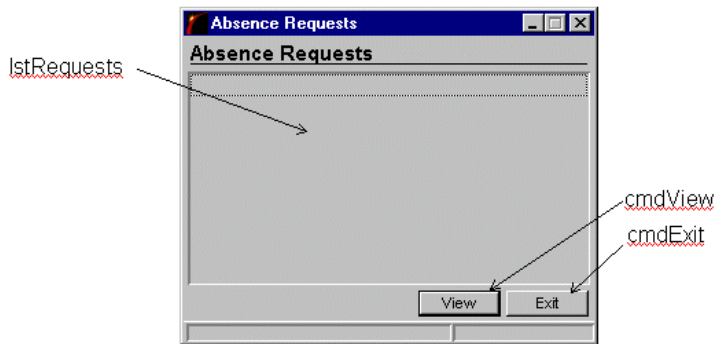
3. Change the default name of the field from its current name (for example, StringWithIcon1007) to 'lstRequests' by modifying its name in the Attributes window.

Again, you assign a prefix to indicate an object's type. In this case, the prefix 'lst' indicates that this is a fixed list field object.



4. Press <Tab> to update the change.

Repeat steps 2-4 of the field naming process and name the form's fields according to the form below. Don't forget to press <Tab> after assigning a new name to each field.



11

When you have named all of the fields entered on the above form, click the form's close box.

## Connecting your project to a database

In this section, you will connect your project to your database.

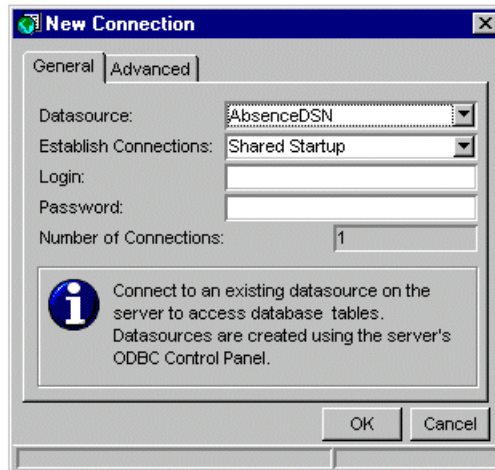
### Creating a connection to the Absence database

If you have completed "Tutorial 2: User Absence form with bound columns (database)", your Microsoft Access database and ODBC data source will already be complete and ready for use

with the Absence Admin project. Since you have already completed these steps you can move directly into FirstClass RAD and begin developing the application.

Creating a new connection:

1. Click New on the Connections tab to display the New Connection form.
2. Select the AbsenceDSN data source from the Datasource selection list.
3. Specify 'Shared Startup' in the Establish Connections list.  
The number of connections will automatically be set to 1.



4. Click OK to save the AbsenceDSN connection.

### **Adding the Requests table to your project**

Once the AbsenceDSN connection has been configured, the Requests table in the database may be added to the project using the Tables tab.

To add the new table:

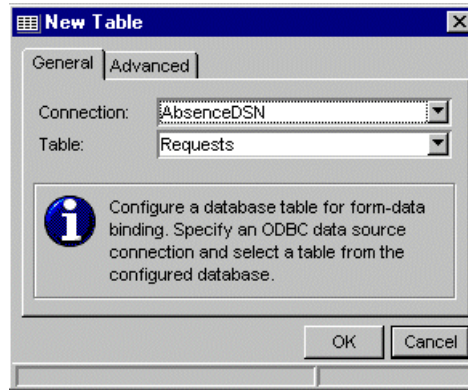
1. Click New on the Table tab to display the Table Configuration form.



2. Select the AbsenceDSN connection from the selection list at the top of the form.

Tables available for that data source will appear in the Table selection list.

3. Select the Requests table from the Tables selection list.



4. Click OK to save the Requests table configuration.

## Programming your project

In this project, the program begins by showing the Absence List form and populating the list with the current absence requests. You do this by creating a query, or database statement, in code and looping through the returned record set.

Programming Sub Main():

1. Select the Module tab in the FirstClass RAD project manager.

2. Select Module1 and click Modify.

The Module1 code window will open.

3. Select 'Main' from the subroutine selection list.

4. Between the “Sub Main()” and “End Sub” commands, type the following code in Sub Main:

```
frmAbsenceList.Show
```

As you will probably remember from previous tutorials, this code simply displays our form interface. In the next few steps you will populate our list from the database.

5. Type the following code, on the next lines in your program:

```
Dim stmt as dbStatement
Dim i as Integer
```

These lines of code use the Dim command to create two new variables you can use in your code. The dbStatement variable will be used to generate a record set and the integer variable will be used as a loop counter as you iterate through our record set.

6. Type the following code, on the next lines in your program:

```
stmt.OpenStatement(AbsenceDSN)
stmt.ExecuteSQL("Select * from Requests")
```

First, this code uses OpenStatement to associate the database statement variable with the connection, AbsenceDSN. AbsenceDSN was the connection that you configured earlier in the section titled Creating a Connection to the Absence Database. The next line of code runs a simple SQL Query to retrieve the record set from the Requests table.

7. The last step of our code loops through the record set returned from the last instructions. Type the following on the next few lines:

```
Do While Not stmt.EOF
    frmAbsenceList.lstRequests(i) =
    stmt.Column("UserName")
    i = i + 1
    stmt.MoveNext
Loop
```

Here you are using a Do...While loop to iterate through the record set. A Do...While loop will iterate until its condition evaluates to FALSE. In this case you are evaluating whether the database statement variable has reached the end of the record set (End Of File - EOF).

Inside the loop you are running instructions to assign the value of each fixed list element to the value of the UserName column of each record. This assignment is accomplished with the first line of code:

```
frmAbsenceList.lstRequests(i) =
    stmt.Column("UserName")
```

After you complete the assignment, you also increment to the list element counter and move to the next database record in preparation for the next iteration. These steps are accomplished with the following code:

```
i = i + 1
stmt.MoveNext
```

Let's see how this algorithm works when the program is running. Imagine that you have the following three records in our Requests table:

RequestID	AbsenceType	StartDate	EndDate	UserName
1	0	8/22/00	8/31/00	John Public
2	3	8/22/00	8/31/00	Mary Doe
3	2	8/22/00	8/31/00	John Class

Our loop, which iterates while "Not stmt.EOF", will execute 3 times—once for each record. Each time you iterate, you perform an assignment to the current list element. The list element is specified by the subscript "(i)" as "i" is increased from 0 to 2 through three iterations.

As the program executes and the variables are evaluated you will have the following three assignments occurring in our loop:

```
frmAbsenceList.lstRequests(0) = "John
Public"
frmAbsenceList.lstRequests(1) = "Mary Doe"
frmAbsenceList.lstRequests(2) = "John
Class"
```

Finally, add one last line of code to assign a hidden key value for each element in the list. When the user clicks an element in the list you will be able to retrieve this numeric value in order to

uniquely identify the record selected. In this case the assignment is made using the primary key in the database table.

The following line of code is entered right after the list value assignment:

```
frmAbsenceList.lstRequests(i).Key =  
stmt.Column("RequestID")
```

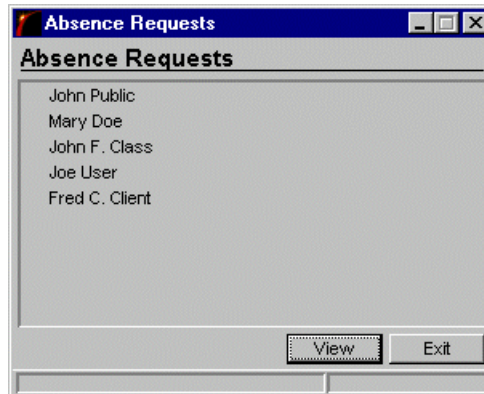
Your completed Sub Main code should look like this:

```
Sub Main()  
  
    frmAbsenceList.Show  
  
    Dim stmt as dbStatement  
    Dim i as Integer  
  
    stmt.OpenStatement(AbsenceDSN)  
    stmt.ExecutesQL("Select * from Requests")  
  
Do While Not stmt.EOF  
  
    frmAbsenceList.lstRequests(i) =  
    stmt.Column("UserName")  
    frmAbsenceList.lstRequests(i).Key =  
    stmt.Column("RequestID")  
  
    i = i + 1  
    stmt.MoveNext  
  
Loop  
  
    stmt.CloseStatement  
  
End Sub
```

### **Running your application**

When you have finished typing in the code, press <Tab> and close the code window. Click the Run to run your application in

Debug Mode. The form will be shown and populated with the records in your database.



## Displaying icons in your list

Next, let us improve our application by showing the FirstClass 'unread' flag with each user name that has their request currently 'Unconfirmed'. An unconfirmed record is one that has its Approved field set to 0.

Modify your Sub Main code and add the change to the Do...While loop by adding an If...Then condition like this:

```
Do While Not stmt.EOF

    frmAbsenceList.lstRequests(i) =
        stmt.Column("UserName")

    If stmt.Column("Approved") = 0
        thenfrmAbsenceList.lstRequests(i).Icon =
            7016
    End If

    i = i + 1

    stmt.MoveNext

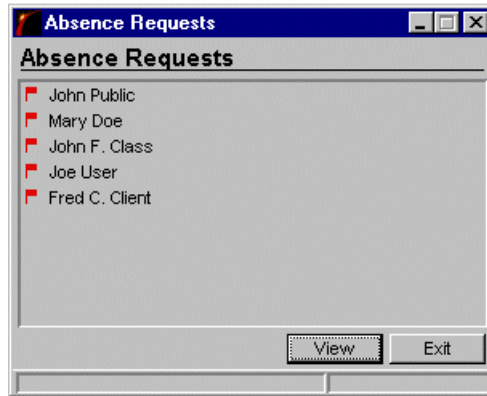
Loop

End loop
```

# 11

The new condition you have added now checks if the current record's Approved value is 0 (Unconfirmed). If it is 0, then you are assigning icon number 7016 (the unread flag icon resource ID) to the icon property of the list element.

Run your program again. All records in your database should be unconfirmed at this point and each record should get a red flag.



In the next section you will add a second form to the project and program the interface to view request details.

## Assembling Project II – adding a second form

In this section you will add a second form to the project and configure it to open from your absence list form. Adding a second form follows the same procedures as adding the first. At the end of this section, you will also be binding this form to the Requests table in the database.

### Adding the Absence Detail form to your project

Adding the Absence Admin Detail Form opens the form in FirstClass RAD. When the form is opened, FirstClass RAD downloads all of the form information into the project database. Once a form has been added, all of its fields may be named and programmed.

To add the form to your project:

1. Click New on the Forms tab.

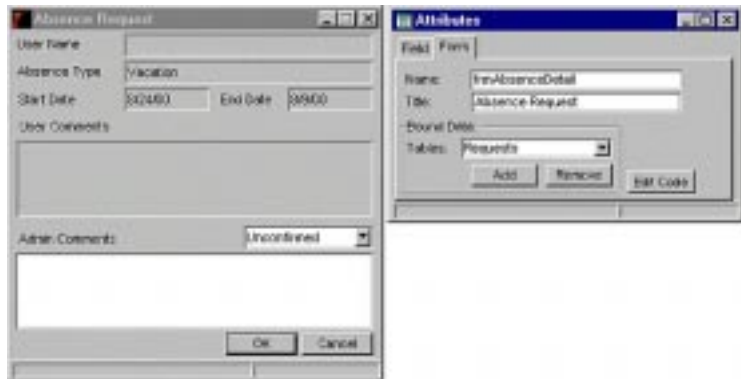
2. Select the 'Absence Admin Detail Form' from the list of forms in the settings document.
3. Click OK to add the form.

The FirstClass RAD Attributes window will open to allow field naming and programming in the next section.

**Naming forms and fields** The next step is to name your forms and fields.

To name your form:

1. Click on the Form tab of the Attributes window to display the default name of the form, the form title, and the form's bound data environment.
2. Edit the form's default name and change it to 'frmAbsenceDetail'.



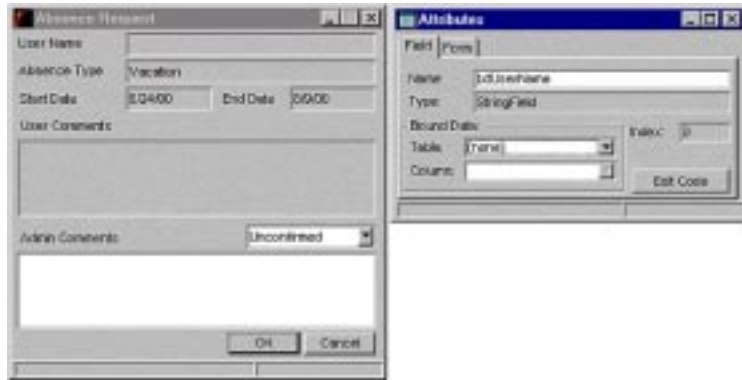
3. Set the form's title to 'Absence Request'.
4. Press <Tab> to update the changes, when you have finished entering the form name and title.

To name fields:

1. Select the Field tab of the Attributes window.
2. Click on the text field immediately right of the guide text 'User Name'.

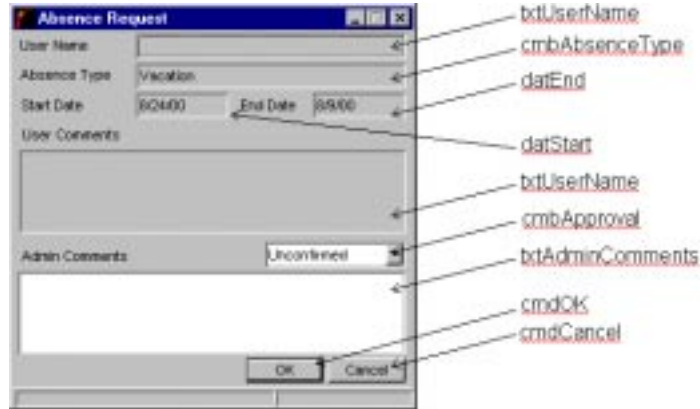
The Attributes window will display the default name of the field.

3. Change the default name of the field from its current name (for example, StringField1007) to 'txtUserName' by modifying its name in the Attributes window.



4. Press <Tab> to update the change, when you have finished entering the field's name.

Repeat steps 2-4 of the field naming process and name the form's fields according to the following diagram. Do not forget to press <Tab> after assigning the name of each field.



When you have named all of the fields on the diagram, close the form by clicking the form's close box.

## Binding fields to columns

Although you didn't bind fields for the Absence List form, you will bind them for this Detail form. In FirstClass RAD you



generally populate lists from a query as you did in the previous section, but when large numbers of fields are associated with a form it is usually safer to use column binding.

### **Adding your table to the data environment**

The first step in binding fields to columns is adding your table to the data environment.

To add a table:

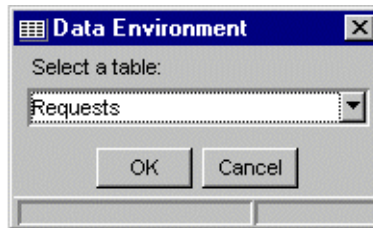
1. Open the form frmAbsenceDetail, either by double-clicking its entry on the Form tab or by selecting it and clicking Modify.
2. Click on the Form tab of the Attributes window.

The form displays form name and data environment information.

3. Click Add, in the Bound Data section of the Attributes window.

The Data Environment dialog will be displayed.

4. Select the Requests table from the drop down list in the Data Environment dialog and click OK.



The Requests table has been added to the data environment of the form and is displayed in the Attributes window. Now, when the Absence Detail form is loaded or shown, a reference to the Requests table data will be created and be associated with the form.

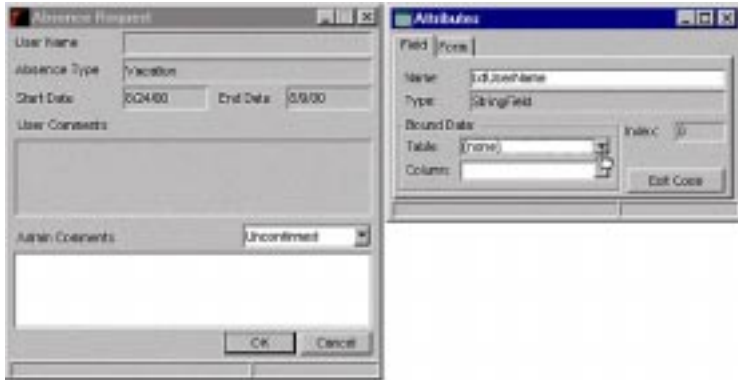
### **Binding fields**

The next step is to bind the table fields. To do this:

1. Select the User Name text field on the form (txtUserName).

2. Select the Requests table with the Table selection list, In the Field tab.

The columns in the table are immediately displayed in the Column selection list.



3. Select the UserName column from the Column selection list.



The txtUserName field is now bound to the UserName database column and data from the database column will be linked with the field when the project is run.

Repeat this process to bind the following fields to the appropriate database columns:

Field name	Bound column
cmbAbsenceType	ABSENCETYPE
txtUserName	USERNAME
datStart	STARTDATE

datEnd	ENDDATE
texComments	COMMENTS
txtAdminComments	ADMINCOMMENTS
cmbApproval	APPROVED

## Programming Project II – integrating your forms

The next step is to program the View button on the Absence List form. When functional, this button will open the detail form and display the selected records' complete details and allow the user to add comments and set approval status.

### Programming buttons on the List form

To program the Click Event on the View button:

1. Open the form frmAbsenceList, either by double-clicking its entry on the Form tab or by selecting it and clicking Modify.
2. Open the code window by double-clicking the View button (cmdView) on the form, or by selecting the button and clicking Edit Code on the Attributes form.
3. Confirm that the Object selection list is set to cmdView and that the Event selection list is set to Click.
4. Add the following code to the Click Event:

```
Load frmAbsenceDetail
frmAbsenceDetail.Requests.Filter
("RequestID=" & Str (1stRequests.Key))
frmAbsenceDetail.Show
```

This code begins by loading the absence detail form. The load command creates a reference to the form in memory and establishes database table associations without actually showing the form. Using the load command you are able to manipulate the data environment of the form before it is displayed to the user.

The second line of code calls the Filter command against the form-bound table Requests. Here, you filter the displayed record to the record that has a RequestID that matches the value you

assigned to the key in the currently selected element of the fixed list. Remember, you must convert the key value to a text string before concatenating it to the text argument for the Filter command. This is done with the Str internal function.

Finally, on the third line of code you show your record with the selected record.

5. Press <Tab> and close the code window, when you have finished typing in the code.

### **Programming the Exit button**

To exit the program, your application will simply unload all of the forms that might be open. To do this, add the following code to the Click event of the cmdExit button.

```
Unload frmAbsenceDetail  
Unload frmAbsenceList
```

## **Programming the buttons on the Absence Detail form**

In this section you will code the Click event on the OK and Cancel buttons on the Absence Detail form.

To program the OK button:

1. Open the form frmAbsenceDetail, either by double-clicking its entry on the Form tab or by selecting it and clicking Modify.
2. Open the code window by double-clicking OK (cmdOK) on the form, or by selecting the button and clicking Edit Code on the Attributes form.
3. Confirm the Object selection list is set to cmdOK and that the Event selection list is set to Click.
4. Add the following code to the Click Event.

```
Requests.Update  
Unload frmAbsenceDetail
```

The first line of code updates all editing changes the user has made to the bound and editable fields on the form; these changes are written back to the database. The second line of

code simply closes the form. You should be able to program the cancel button now.

Use the Unload command to close the form and discard any changes made to the record.

## Running your application

Your application is now complete. Let's test to make sure the application is functioning properly.

1. Click Run to run your application in Debug Mode.

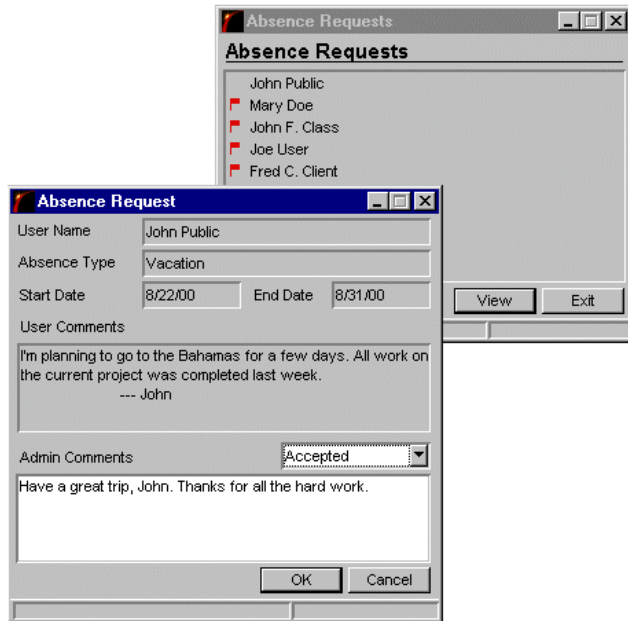
You will see your absence list and associated red flags.

2. Select a record in the list by clicking it.

3. Click View.

The detail form will be shown with the selected record.

4. Approve the selected record, type in a comment and click the OK button.



Your changes have now been written to the database.

5. Open the record a second time to confirm that your edits have been saved.

If you quit and rerun the application, you will not have an unread flag on the record because you approved it.

## Building and installing your application

After completing the project's programming and testing, you are ready to install the application on your FirstClass Intranet Server.

### Building your project

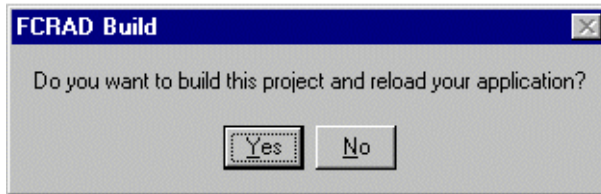
When a project is ready to be installed, the build process creates a compiled form of the project that the FirstClass Intranet Server uses to host the application.

To build your project:

1. Click Build on the Project tab.

A message box will appear and confirm that you are interested in building and reloading your application.

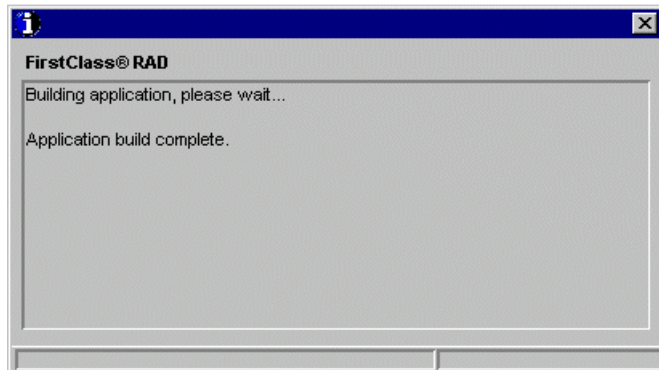
2. Click Yes to confirm the build.



The build process creates an FCX, or FirstClass executable, file. Our project is named 'absenceadmin', so the resulting FCX file created is called absenceadmin.fcx.

This file is written to the FirstClass RAD folder in your FirstClass Post Office. The FirstClass RAD folder contains built applications that are loaded by your FirstClass server at

startup. When the build process is complete, you should see the following report window:



When the application build is complete, close this window.

An application icon has been created in the 'Built Applications' folder in the FirstClass RAD folder. You may double-click this icon to launch your application from the FirstClass client. If you want to install additional application icons, follow the steps in "Installing additional application icons" on page 85.

# 11

## Extra practice

Now that you have completed the Absence Admin application, here are a few other ideas you can try for an extra challenge.

1. Add the following code to the DoubleClick event of lstRequests to view a record if the user double-clicks it:
 

```
cmdView.Click
```
2. Add code to the cmdOK button that sends an email message to the requesting user indicating the approval or rejection of her request.
3. Add code to the Click event of cmdView that turns off the flag icon automatically when the user views a record.
4. Use an ORDER BY clause in your SQL statement to sort the requests by the Approved field. Make all of the Unconfirmed records sort to the top of the list. See your Microsoft Access

Extra practice

documentation for help with the SELECT and ORDER BY commands.



# Appendix



## FirstClass Designer field attributes

As a rule, field attributes apply to all fields the same way in both FirstClass Designer and FirstClass RAD. There are, however, certain exceptions to this rule: certain fields having special attributes when used in RAD. The following tables show all of the field types, field methods, and their corresponding field attributes, as they apply in FirstClass Designer.

For complete details on field types, field attributes, and field methods, see our online help.

The following tables outline which field attributes and field methods (vertical) are associated with which FirstClass Designer fields (horizontal):

Table of Graphics fields

		Picture	Icon	Line	Rectangle	Round Rectangle	Oval
<b>Field attributes</b>	<b>Data type</b>						
Admin	Boolean						
Bold	Boolean	X	X	X	X	X	X
Border	Boolean	X	X	X	X	X	X
Bottom	Boolean	X	X	X	X	X	X
Caption	Boolean						
Center	Boolean	X	X	X	X	X	X
Color	Boolean	X	X	X	X	X	X
Condense	Boolean	X	X	X	X	X	X
Datatype	Special	X	X	X	X	X	X
DoubleClick	Boolean	X	X	X	X	X	X

		Picture	Icon	Line	Rectangle	Round Rectangle	Oval
Field attributes	Data type						
Editable	Boolean	X	X	X	X	X	X
Extend	Boolean	X	X	X	X	X	X
FullJust	Boolean	X	X	X	X	X	X
Grey	Boolean	X	X	X	X	X	X
Hidden	Boolean	X	X	X	X	X	X
Icon	Integer		X				
Index	Integer						
Italic	Boolean	X	X	X	X	X	X
Key	Any						
ListCount	Integer						
Name	String						
NoOpenSpace	Boolean						
Outline	Boolean	X	X	X	X	X	X
Password	Boolean	X	X	X	X	X	X
Protected	Boolean	X	X	X	X	X	X
RJust	Boolean	X	X	X	X	X	X
Selectable	Boolean	X	X	X	X	X	X
Selected	Boolean	X	X	X	X	X	X
Shadow	Boolean	X	X	X	X	X	X
Text	Boolean	X	X	X	X	X	X
Transparent	Boolean	X	X	X	X	X	X
Underline	Boolean	X	X	X	X	X	X

		Picture	Icon	Line	Rectangle	Round Rectangle	Oval
<b>Field attributes</b>	<b>Data type</b>						
Wrap	Boolean	X	X	X	X	X	X
<b>Field methods</b>							
AddItem	N/A						
Change	N/A	X	X				
Click	N/A	X	X	X	X	X	X
DblClick	N/A	X	X	X	X	X	X
GotFocus	N/A	X	X	X	X	X	X
LostFocus	N/A	X	X	X	X	X	X
RemoveItem	N/A						
SetFocus	N/A	X	X	X	X	X	X

Table of Text &amp; Numbers fields

		Editable Text	Guide Text	String with Icon	Marquee	Number	Group Box	Time Period
<b>Field attributes</b>	<b>Data type</b>							
Admin	Boolean							
Bold	Boolean	X	X	X	X	X	X	X
Border	Boolean	X	X	X	X	X	X	X
Bottom	Boolean	X	X	X	X	X	X	X
Caption								
Center	Boolean						X	
Color	Boolean	X	X	X	X	X	X	X
AddItem	Boolean	X	X	X	X	X	X	X
Condense	Special	X	X	X	X	X	X	X
Datatype	Boolean	X	X	X	X	X	X	X
DoubleClick	Boolean	X	X	X	X	X	X	X
Editable	Boolean	X	X	X	X	X	X	X

A

		Editable Text	Guide Text	String with Icon	Marquee	Number	Group Box	Time Period
<b>Field attributes</b>	<b>Data type</b>							
Extend	Boolean	X	X	X	X	X	X	X
FullJust	Boolean	X	X	X	X	X	X	X
Grey	Boolean	X	X	X	X	X	X	X
Hidden	Integer	X	X	X	X	X	X	X
Icon	Integer			X				
Index	Boolean							
Italic	Any	X	X	X	X	X	X	X
Key	Integer							
ListCount	String							
Name	Boolean							
NoOpenSpace	Boolean							
Outline	Boolean	X	X	X	X	X	X	X
Password	Boolean	X	X	X	X	X	X	X
Protected	Boolean	X	X	X	X	X	X	X
RJust	Boolean	X	X	X	X	X	X	X
Selectable	Boolean	X	X	X	X	X	X	X
Selected	Boolean	X	X	X	X	X	X	X
SellLength	Boolean	X	X					
SelStart	Boolean	X	X					
Shadow	Boolean	X	X	X	X	X	X	X
Text	Boolean	X	X	X	X	X	X	X
Transparent	Boolean	X	X	X	X	X	X	X
Underline	Boolean	X	X	X	X	X	X	X
Value	Boolean					X	X	X
Wrap	Boolean	X	X	X	X	X	X	X
<b>Field methods</b>								
AddItem	N/A							
Change	N/A	X	X	X	X	X		X

		Editable Text	Guide Text	String with Icon	Marquee	Number	Group Box	Time Period
Field attributes	Data type							
Click	N/A	X	X	X	X	X	X	X
DbClick	N/A	X	X	X	X	X	X	X
GotFocus	N/A	X	X	X	X	X	X	X
LostFocus	N/A	X	X	X	X	X	X	X
RemoveItem	N/A							
SetFocus	N/A	X	X	X	X	X	X	X

Table of Buttons fields, Extensions, and Tab Controls fields

		Checkbox	Radio Group	Radio Button	Command Button	URL Button	Group Box	Progress Bar	Expanding List	File Viewer	Fixed List	Single Tab
Field attributes	Data type											
Admin	Boolean											
Bold	Boolean	X	X	X	X	X		X	X	X	X	
Border	Boolean	X	X	X	X	X		X	X	X	X	
Bottom	Boolean	X	X	X	X	X		X	X	X	X	
Caption	Boolean	X	X	X	X			X				X
Center	Boolean	X	X	X	X	X		X	X	X	X	
Color	Boolean	X	X	X	X	X		X	X	X	X	
AddItem	Boolean	X	X	X	X	X		X	X	X	X	
Condense	Special	X	X	X	X	X				X		
Datatype	Boolean	X	X	X	X	X				X		
DoubleClick	Boolean	X	X	X	X	X				X		
Editable	Boolean	X	X	X	X	X		X	X	X	X	
Extend	Boolean	X	X	X	X	X		X	X	X	X	

A

		Checkbox	Radio Group	Radio Button	Command Button	URL Button	Group Box	Progress Bar	Expanding List	File Viewer	Fixed List	Single Tab
<b>Field attributes</b>	<b>Data type</b>											
FullJust	Boolean	X	X	X	X	X		X	X	X	X	
Grey	Boolean	X	X	X	X	X		X	X	X	X	
Hidden	Integer	X	X	X	X	X				X		
Icon	Integer				X					X		X
Index	Boolean									X		
Italic	Any	X	X	X	X	X				X		
Key	Integer									X		
ListCount	String							X	X	X	X	
Name	Boolean							X	X	X	X	
NoOpenSpace	Boolean							X	X	X	X	
Outline	Boolean	X	X	X	X	X			X			
Password	Boolean	X	X	X	X	X			X		X	
Protected	Boolean	X	X	X	X	X		X	X	X	X	
RJust	Boolean	X	X	X	X	X			X		X	
Selectable	Boolean	X	X	X	X	X					X	
Selected	Boolean	X	X	X	X	X						
SelLength	Boolean	X	X	X	X	X				X		
SelStart	Boolean	X	X	X	X	X						
Shadow	Boolean	X	X	X	X	X			X			
Text	Boolean	X	X	X	X	X		X	X	X	X	
Transparent	Boolean	X	X	X	X	X		X	X	X	X	
Underline	Boolean	X	X	X	X	X		X	X	X	X	
Value	Boolean									X		
Wrap	Boolean							X	X	X	X	
<b>Field methods</b>								<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>	
AddItem	N/A											



		Checkbox	Radio Group	Radio Button	Command Button	URL Button	Group Box	Progress Bar	Expanding List	File Viewer	Fixed List	Single Tab
Field attributes	Data type											
Change	N/A	X	X	X				X	X	X	X	
Click	N/A	X	X	X	X	X		X	X	X	X	
DbClick	N/A	X	X	X	X	X			X	X	X	
GotFocus	N/A	X	X	X	X	X		X	X	X	X	
LostFocus	N/A	X	X	X	X	X		X	X	X	X	
RemoveItem	N/A											
SetFocus	N/A							X	X	X	X	

Table of Lists fields and Selectors fields

		Static Selection List	Editable Selection List	Date Selector	Font Selector	Color Selector
Field attributes	Data type					
Admin	Boolean					
Bold	Boolean	X	X	X		
Border	Boolean	X	X	X		
Bottom	Boolean	X	X	X		
Caption	Boolean					
Center	Boolean	X	X	X		
Color	Boolean	X	X	X		
Condense	Boolean	X	X	X		
Datatype	Special	X	X	X		
DoubleClick	Boolean	X	X	X		
Editable	Boolean	X	X	X		

A

		Static Selection List	Editable Selection List	Date Selector	Font Selector	Color Selector
<b>Field attributes</b>	<b>Data type</b>					
Extend	Boolean	X	X	X		
FullJust	Boolean	X	X	X		
Grey	Boolean	X	X	X		
Hidden	Boolean	X	X			
Index	Integer			X		
Italic	Integer	X	X			
Key	Boolean					
List	Any	X	X			
Name	Integer					
NoOpenSpace	String			X		
Outline	Boolean	X	X	X		
Password	Boolean	X	X	X		
Protected	Boolean	X	X	X		
RJust	Boolean	X	X	X		
Selectable	Boolean	X	X	X		
Selected	Boolean	X	X	X		
Shadow	Boolean	X	X	X		
Transparent	Boolean	X	X	X		
Underline	Boolean	X	X	X		
Value	Boolean	X		X		
Wrap	Boolean	X	X	X		
<b>Field methods</b>						
AddItem	N/A	X	X			
Change	N/A					
Click	N/A					
DblClick	N/A					
GotFocus	N/A					

		Static Selection List	Editable Selection List	Date Selector	Font Selector	Color Selector
<b>Field attributes</b>	<b>Data type</b>					
LostFocus	N/A					
RemoveItem	N/A	X	X			
SetFocus	N/A	X	X	X		

**A**



# FirstClass RAD error messages

FirstClass RAD error messages are loosely divided into categories based on the error message number.

## Core error messages (1 – 107)

**1**

*Message:* **Syntax error.**

You have made a mistake in the code syntax. You will get a line number where the error occurs.

**2**

*Message:* **Memory allocation failure.**

Your request exceeds the available amount of memory on your system.

**3**

*Message:* **Program entry point not found, no Sub Main().**

Sub Main( ) is not found anywhere in the program code.

**4**

*Message:* **Unbalanced parentheses.**

You have left out a parenthesis (open or close) in the program code.

**5**

*Message:* **Equals sign expected.**

You have left out an equals sign (=) in the appropriate place in the program code. This can occur in the assignment statements or when assigning variables.

**6**

*Message:* **Duplicate label.**

**B**

You have created two labels with the same name in the program code.

**7**

*Message: **Undefined label.***

You haven't defined a label in the program code.

**8**

*Message: **THEN expected.***

You have left out a THEN command.

**9**

*Message: **TO expected.***

You have left out a TO command.

**10**

*Message: **NEXT without FOR.***

You have left out a FOR command.

**11**

*Message: **Invalid identifier name.***

You have an identifier name with invalid characters.

**12**

*Message: **AS expected.***

There is a DIM command without AS.

**13**

*Message: **Invalid variable/type/sub/function redefinition.***

You have set a variable, type, sub, or function twice or incorrectly.

**14**

*Message: **Undeclared identifier.***

You have used an identifier without declaring it.

**15**

*Message: **Type mismatch error.***

You have an assignment between two types that are not the same.

**16**

*Message: **Divide by zero error.***

You have a calculation in your code that is divisible by zero. This is an impossible calculation.

**17**

*Message:* **END IF expected.**

You have left out a required END IF command.

**18**

*Message:* **END IF without IF.**

You have left out a required IF command.

**19**

*Message:* **ELSE without IF.**

You have left out a required IF command.

**20**

*Message:* **WEND without WHILE.**

You have left out a required WHILE command.

**21**

*Message:* **WEND expected.**

You have left out a required WEND command.

**22**

*Message:* **Incorrect number of indices for array.**

You have used too few or too many indices in defining your array.

**23**

*Message:* **Identifier expected.**

You have left out a required identifier name.

**24**

*Message:* **Array index out of range.**

You have accessed a number in an array outside of the defined array boundaries.

**25**

*Message:* **Variable is not an array.**

You have not defined the variable as an array.

**26**

*Message:* **Redefinition of a constant not allowed.**

You have two constants with the same name.

**B**

**27**

Message: **END SELECT expected.**

You have left out a required END SELECT command.

**28**

Message: **END SELECT without SELECT CASE.**

You have left out a required SELECT CASE command.

**29**

Message: **CASE expected.**

You have left out a required CASE command. CASE is a subset of SELECT CASE.

**30**

Message: **Incorrect number of arguments to subroutine.**

You have called a subroutine with an incorrect number of arguments.

**31**

Message: **Invalid range for an array.**

You have defined an array with an incorrect range.

For example, you can't have negative numbers in an array. The lower bound number in an array must be smaller than the upper bound number.

**32**

Message: **Keywords may not be used as variables.**

You have used a keyword a variable. Keywords are part of the program language and cannot be used to name variables.

**33**

Message: **memory allocation error; array size too large.**

You have created an array size that is larger than the available memory on your system.

**34**

Message: **Unknown method reference.**

You have referenced a method that does not exist.

**35**

Message: **Undefined object.**

You have an undefined object in the code.



**36**

*Message:* **Invalid object reference.**

You have referenced an object that does not exist.

**37**

*Message:* **IF expected.**

You have left out a required IF command in an IF...THEN ELSE statement.

**38**

*Message:* **Expression expected.**

You have not entered the expression in the code.

**39**

*Message:* **ELSEIF without IF.**

You have left out an IF command in an IF...ELSEIF statement.

**40**

*Message:* **ELSEIF after ELSE.**

You have placed the ELSEIF command after the ELSE command. ELSE must be the last clause.

**41**

*Message:* **Expected end of statement or new line.**

You have placed several statements on one line.

**42**

*Message:* **Data type expected.**

You have not properly defined a data type.

**43**

*Message:* **Invalid counter for NEXT statement.**

You have used an incorrect counter for a loop.

**44**

*Message:* **FOR without NEXT.**

You have left out the NEXT command in a FOR...NEXT statement.

**45**

*Message:* **No statements allowed between SELECT CASE and first CASE.**

**B**

You have code in between the SELECT CASE and first CASE commands.

**46**

*Message:* **Expected expression, IS, or ELSE.**

You have created a CASE expression without IS or ELSE commands.

**47**

*Message:* **Expected comparison operator after IS.**

You have not entered a comparison operator after the IS command.

**48**

*Message:* **CASE without SELECT CASE.**

You have left out a SELECT CASE command appears.

**49**

*Message:* **LOOP without DO.**

You have left out a DO command in a DO...LOOP statement.

**50**

*Message:* **EXIT DO allowed only within DO LOOP.**

You have placed an EXIT DO command outside of a DO LOOP command pair.

**51**

*Message:* **DO without LOOP.**

You have left out a LOOP command in a DO...LOOP statement.

**52**

*Message:* **Expected UNTIL or WHILE.**

You have left out an UNTIL or WHILE command.

**53**

*Message:* **EXIT FOR allowed only within FOR NEXT.**

You have placed an EXIT FOR command outside a FOR NEXT command.

**54**

*Message:* **Equal sign or AS expected.**

There is a DIM command without an equal sign (=) or AS.

**55**

*Message: **Error in label definition.***

You have incorrectly defined a label.

**56**

*Message: **Duplicate OPTION BASE statement.***

You have duplicated an OPTION BASE command.

**57**

*Message: **BASE expected.***

You have used an OPTION command without a BASE command.

**58**

*Message: **0 or 1 expected.***

You have used a BASE option other than 0 or 1.

**59**

*Message: **Integer or constant value expected.***

You have specified the number of characters in a string, but did not provide actual integer numbers in a TYPE statement.

**60**

*Message: **AS without DIM, REDIM, or TYPE.***

You have omitted either the DIM, REDIM, or TYPE commands with the AS command when defining variables in a code statement.

**61**

*Message: **TO without preceding statement.***

You have the TO command on a line without any other commands.

**62**

*Message: **TYPE expected.***

You have used the ENDTYPE command without the TYPE command.

**63**

*Message: **TYPE without members.***

You have established user-defined types with the TYPE command but have not defined any members.

**64**

*Message: **TYPE without END TYPE.***

**B**

You have used the TYPE command without the corresponding END TYPE command.

**65**

*Message:* **END TYPE without TYPE.**

You have used the END TYPE command without the corresponding TYPE command.

**66**

*Message:* **End of line required after ELSEIF...THEN.**

There is no return at the end of the line.

**67**

*Message:* **Variables and function calls not allowed in the expression.**

You have entered a variable or function call when a constant is expected.

**68**

*Message:* **The range for fixed length strings is 1 - 65536.**

You have defined a string length longer than the range.

**69**

*Message:* **STEP 0 not allowed.**

You have entered STEP 0 as a loop increment. Program will not loop.

**70**

*Message:* **RETURN without GOSUB.**

You have used the RETURN command without the corresponding GOSUB command.

**71**

*Message:* **ERROR expected.**

You have used the ON command without the corresponding ON ERROR command.

**72**

*Message:* **GOTO or RESUME expected.**

You have used the ON ERROR command without the GOTO or RESUME commands.

**73**

*Message:* **NEXT expected.**

You have used the FOR command without the corresponding NEXT command.

**74**

*Message:* **Undefined data type.**

You have not defined the data type in the program code.

**75**

*Message:* **RESUME without enabled error handling.**

You have used the RESUME command without the corresponding ON ERROR command.

**76**

*Message:* **RESUME without error.**

You have used the RESUME command without defining the error type.

**77**

*Message:* **Invalid variable name.**

You have incorrectly defined the variable name.

**78**

*Message:* **Single-line nested IF..THEN not supported.**

You have two IF..THEN commands on the same line.

**79**

*Message:* **Identifier too long.**

You have defined an identifier longer than the 128 character maximum.

**80**

*Message:* **Invalid use of system function.**

You have used a function without using the return value.

For example, in an IF..THEN statement or in an assignment statement.

**81**

*Message:* **Element identifier required.**

You have referred to an array without specifying a particular element array.

**82**

**B**

*Message:* **SUB without END SUB.**

You have used the SUB command without the corresponding END SUB command.

**83**

*Message:* **FUNCTION without END FUNCTION.**

You have used the FUNCTION command without the corresponding END FUNCTION command.

**84**

*Message:* **Nested procedures not allowed.**

You have defined two or more procedures within same procedure.

**85**

*Message:* **No statements allowed outside procedures.**

You have statement(s) outside of procedures.

**86**

*Message:* **Illegal END SUB/FUNCTION.**

You have closed your command incorrectly.

**87**

*Message:* **Incorrect object method.**

You have entered an object method incorrectly.

For example, `Form.loaf` instead of `Form.load`.

**88**

*Message:* **No labels allowed outside of procedures and functions.**

You have one or more labels outside of procedures or functions.

**89**

*Message:* **Invalid procedure call.**

You have called a subroutine or function incorrectly. Check where you have defined your subroutine or function.

**90**

*Message:* **Illegal resizing operation in REDIM.**

You have entered the starting number of an array as different from the original variable, or the number of dimensions of an array doesn't match.

**91**

*Message:* **Cannot change the number of dimensions in an array.**

You have tried to change the number of dimensions in an array.

**92**

*Message:* **EXIT FUNCTION not allowed in SUB.**

You have used an EXIT FUNCTION in SUB.

**93**

*Message:* **EXIT SUB not allowed in the function.**

You have used an EXIT SUB in a function..

**94**

*Message:* **Overflow.**

The assignment has exceeded the range of the variable.

**95**

*Message:* **Invalid inside procedure.**

You have used either the TYPE or OPTION keywords in a procedure.

**96**

*Message:* **Invalid outside procedure.**

You have written code outside of a procedure.

**97**

*Message:* **Invalid attribute value.**

You have assigned an incorrect value to an attribute.

**98**

*Message:* **Unterminated quoted string.**

You have forgotten to put an end quote in a literal string before the line return.

**99**

*Message:* **Expected function or variable.**

You have incorrectly called a method without specifying a variable name.

**100**

*Message:* **Incorrect source database version.**

The source code was generated by a Rapid Application Developer (RAD) version earlier than 1.1.

**101**

*Message:* **Empty sub or function.**

You haven't put code in the SUB or function.

**102**

*Message:* **Labels in Declarations not allowed.**

You have put a label in a declaration section of your code.

**103**

*Message:* **File must be opened to perform this operation.**

You have tried to perform an operation on a closed file.

**104**

*Message:* **File open failed.**

Your file does not exist, is corrupt, or is in use.

**105**

*Message:* **Unable to position file pointer.**

You have gone beyond the length of the file or given the file a negative (-) pointer number.

**106**

*Message:* **Box array index out of range.**

You have referred to a box element that is out of range or less than zero (0).

**107**

*Message:* **Database column does not support NULL.**

You have entered a null value for the database column.

## **Database error messages (500 – 509)**

**500**

*Message:* **Data source connection has not been established.**

You have not established the data source connection for your program.

**501**



**Message: Invalid cursor state.**

You have attempted a database operation on a statement that is closed.

**502****Message: ODBC error.**

An ODBC error was returned.

**503****Message: Undefined column.**

You have tried to access a column that doesn't exist.

**504****Message: Undefined attribute.**

You have tried to access an attribute that doesn't exist.

**505****Message: Unsupported data type.**

You have tried to assign data to a field that doesn't support that data type.

**506****Message: Statement must be closed before reopening on a connection.**

You have tried to open a statement twice on the same connection.

**507****Message: Connection must be closed before connecting to a new data source.**

You have tried to connect to a new data source before closing your connection.

**508****Message: Attribute is read-only at runtime.**

You have tried to set an attribute for a database that can't be modified.

**509****Message: ODBC warning.**

There is a problem with your ODBC operation. The operation will still be completed.

## **FirstClass error messages (1000 – 1018)**

### **1000**

*Message: **Attribute or method not found.***

You have tried to call an attribute or method that doesn't exist.

### **1001**

*Message: **Application not found.***

You have tried to run an application that doesn't exist.

### **1002**

*Message: **Application entry point not found.***

You have left out the Sub Main () procedure in the code module. The application doesn't know where to start.

### **1003**

*Message: **All ODBC database connections in use.***

You have exceeded the ODBC driver connection limit. Make sure your connection is set to Shared.

### **1004**

*Message: **You may only run one copy of this application at a time.***

You have tried to run more than one copy of the application.

The application is configured to run only once. You can change the run number on the Options > Project Attributes form and rebuild the application.

### **1005**

*Message: **All of this application's sessions are currently in use.***

You have exceeded the number of sessions set on the Options > Project Attributes form.

### **1006**

*Message: **Form ID number invalid.***

You have set a form ID number to less than 1.

### **1007**

*Message: **Invalid file format.***

You have tried to load an FCX file, which is not recognized by the RAD version you are running.

For example, if you tried to run a FirstClass RAD 2.0 application on an earlier version of FirstClass RAD.

**1008**

*Message:* **Unable to read from file.**

You have tried to load a corrupted FCX file. The error is found when you try to load the application.

**1009**

*Message:* **Unable to write to file.**

You have tried to load a corrupted FCX file. The error is found when you try to build the application.

**1010**

*Message:* **ODBC data source connection failed.**

You have configured a connection that isn't functional. For example, the user ID or password is invalid.

**1011**

*Message:* **Email message recipient not specified.**

The email To attribute is not filled by the Send attribute called.

**1012**

*Message:* **Email message sender not specified.**

The email From attribute is not filled by the Send attribute called.

**1013**

*Message:* **License limit reached; maximum concurrent application sessions exceeded.**

You have exceeded your license limit for concurrent application sessions. This error occurs only on 5.5 non Gold FirstClass servers and earlier.

**1014**

*Message:* **File copy failed.**

The COPYFILE keyword failed. For example, the path and directory are wrong.

**1015**

*Message:* **Field is not an expanding database list.**

You have tried to refer to an element in a field that is not the correct type.

**B**

**1016**

*Message:* **Attribute or method not available for this field.**

You have tried to use a method specific to a field type that is not supported.

For information on which field attributes and field methods are associated with which FirstClass Designer fields, see "FirstClass Designer field attributes" on page 145 in the Appendix.

# Index

## A

- About this book 5
- Adding a form to your project
  - Tutorial 1 67
- Adding a new form to your project
  - Tutorial 2 93
- Adding a new form to your settings document
  - steps to creating FirstClass settings file and resources 45
  - Tutorial 1 64
  - Tutorial 2 90
  - Tutorial 3 116
- Adding fields to the Absence Detail form
  - Tutorial 3 120
- Adding fields to your form
  - Tutorial 1 65
  - Tutorial 2 91
  - Tutorial 3 117
- Adding the Absence Detail form to your project
  - Tutorial 3 132
- Adding the Absence List form to your project
  - Tutorial 3 122
- Adding the Requests Table to your project
  - Tutorial 2 101
- Adding the Requests table to your project
  - Tutorial 3 126
- Adding your table to the data environment
  - Tutorial 2 102
  - Tutorial 3 135
- Administrator tasks 9
- Appendix 143
- Application Monitor 58
- Assembling Project II – adding a second form
  - Tutorial 3 132
- Assembling the project

- Tutorial 1 64
- Tutorial 2 90
- Tutorial 3 116

- Assigning attributes
  - Tutorial 1 66
  - Tutorial 2 92
  - Tutorial 3 118
- Assigning field attributes
  - Tutorial 3 120
- Assigning the 'txtUserName' field
  - Tutorial 2 111

## B

- BASIC programming and FirstClass RAD 32
- BASIC programming overview 30
- Binding fields
  - Tutorial 3 135
- Binding fields to columns
  - Tutorial 2 102
  - Tutorial 3 134
- Binding form fields
  - Tutorial 2 104
- Box arrays 27
- Branding your application 22
- Building and installing your application
  - Tutorial 1 84
  - Tutorial 2 112
  - Tutorial 3 140
- Building your database table
  - Tutorial 2 97
- Building your project
  - Tutorial 1 84
  - Tutorial 2 112
  - Tutorial 3 140

## C

- Calling your subroutine

- Tutorial 1 76
- Code modules 32
- Coding concepts 25
- Coding conventions
  - used in code 31
  - used in this book 7
- Columns 34
- Combining multiple lines of code 31
- Configuring ODBC
  - Tutorial 2 98
- Confirming an inserted record with the Update command
  - Tutorial 2 108
- Connecting a project to a database
  - Tutorial 2 100
- Connecting your project to a database
  - Tutorial 3 125
- Connection tab 18
- Connections 34
- Continuing code on a new line 31
- Core error messages (1 – 107) 155
- Creating a bound-column database application 50
- Creating a connection to the Absence database
  - Tutorial 3 125
- Creating a Microsoft Access database
  - Tutorial 2 97
- Creating a new project
  - Tutorial 1 66
  - Tutorial 2 93
  - Tutorial 3 121
- Creating a new record
  - Tutorial 2 107
- Creating a nonbound- column database application 52
- Creating a nondatabase application 49
- Creating a subroutine
  - Tutorial 1 74
- Creating your FirstClass RAD applications 10, 43
- Creating your FirstClass settings file and resources 43
- Creating your forms 44

- Creating your settings document 43

## D

- Database application basics 33
- Database error messages (500 – 509) 166
- Designer field attributes 48
- Designer field categories 47
- Designer form - Database Hitlist 45
- Designing your forms 44
- Displaying icons in your list
  - Tutorial 3 131
- Distributing completed applications 22
- Documentation conventions 6
- Drawing fields on your form 46

## E

- Event procedures 31
- Extra practice 87, 113
  - Tutorial 1 87
  - Tutorial 2 113
  - Tutorial 3 141

## F

- Field array for 100 checkboxes 26
- Field arrays 26
- FirstClass Application Monitor 54
- FirstClass Designer field attributes 145
- FirstClass error messages (1000 – 1018) 168
- FirstClass RAD concepts 15
- FirstClass RAD database application flowchart 12
- FirstClass RAD environment 4
- FirstClass RAD error messages 8, 155
- FirstClass RAD nondatabase application flowchart 11
- FirstClass specific information 25
- Fixed and expanding lists 28
- Flowcharts of FirstClass RAD applications 10

- Form modules 32
- Form properties 46
- Formatting output
  - Tutorial 1 78
- Formatting output (continued)
  - Tutorial 1 82
- Forming compound strings
  - Tutorial 1 73
- Forms tab 17

## **G**

- General steps for creating a FirstClass RAD application 12

## **H**

- How FirstClass RAD forms and databases work together 19

## **I**

- Inserting a new temporary record
  - Tutorial 2 107
- Installation and startup 37
- Installing additional application icons
  - Tutorial 1 85
- Installing FirstClass RAD 37
- Introduction 1

## **L**

- Loading and unloading an application 58

## **M**

- Mac® OS
  - installing on 38
- Maintaining FirstClass RAD 55
- Managing projects 15
- Menu items 6
- Module tab 17
- Monitoring FirstClass RAD applications 57

## **N**

- Naming forms and fields
  - Tutorial 1 68
  - Tutorial 2 94
  - Tutorial 3 123, 133
- New in this release 8

## **O**

- Object Oriented programming 31
- ODBC drivers 7
- ODBC error messages 8
- Overview 3

## **P**

- Planning and creating FirstClass RAD applications 39
- Planning questions 41
- Planning your FirstClass RAD applications
  - administrator task section 9
  - planning section 41
- Preparing your FirstClass settings document and resources 10
- Programming buttons on the List form 137
- Programming Project II – integrating your forms
  - Tutorial 3 137
- Programming the buttons on the Absence Detail form
  - Tutorial 3 138
- Programming the Cancel button
  - Tutorial 2 110
- Programming the Exit button
  - Tutorial 1 81
  - Tutorial 3 138
- Programming the Refresh button
  - Tutorial 1 80
- Programming the Submit button
  - Tutorial 2 109
- Programming your project
  - Tutorial 1 70

Tutorial 2 105  
Tutorial 3 127  
Project tab 16

## R

Recording application information 54  
Retrieving disk use information  
    Tutorial 1 72  
Running FirstClass server  
    applications 23  
Running the application  
    Tutorial 1 81  
Running the Disk Usage utility in  
    FirstClass RAD  
    Tutorial 1 71  
Running your application  
    Tutorial 3 130, 139

## S

Setting field attributes 48  
    Tutorial 1 65  
    Tutorial 2 91  
    Tutorial 3 118  
Setting the Absence Approval list  
    Tutorial 3 121  
Setting the Absence Type list  
    Tutorial 3 120  
Setting the Absence type list  
    Tutorial 2 92  
Showing the form with Sub Main()  
    Tutorial 1 70  
    Tutorial 2 105  
Statements 34  
Statistics events 28  
System requirements 37

## T

Table of Buttons fields, Extensions, and  
    Tab Controls fields 149  
Table of Graphics fields 145  
Table of Lists fields and Selectors

    fields 151  
Table of Text & Numbers fields 147  
Table tab 18  
Tutorial materials  
    Tutorial 1 63  
    Tutorial 2 90  
    Tutorial 3 116  
Tutorial 1  
    Disk Usage utility (nondatabase)  
        Tutorial 1 63  
Tutorial 2  
    User Absence form with bound  
        columns (database)  
        Tutorial 2 89  
Tutorial 3  
    Administrator Absence form with  
        non-bound columns (database)  
        Tutorial 3 115  
Tutorials 61

## U

Uses for FirstClass RAD applications 4  
Using forms and resources 18  
Using source code 21  
Using the ODBC Driver Manager  
    Tutorial 2 98

## V

Variables 7

## W

What you should already know 6  
Who should read this book 6  
Windows 37  
Working in the Project Manager 49  
Working with databases 19  
Working with forms and fields  
    Tutorial 1 67  
    Tutorial 2 93