Data Collection with PDAs: A Computing Laboratory Application

by

Di Cao

Master of Computer Science, University of North Dakota, 2002

A Project

Submitted to the Graduate Faculty

of the

University of North Dakota

in partial fulfillment of the requirements

for the degree of

Master of Science

Grand Forks, North Dakota

August 2002

This report, submitted by Di Cao in partial fulfillment of the requirements for the Degree of Master of Science from the University of North Dakota, has been read by the Faculty Advisor under whom the work has been done and is hereby approved.

Thomas Wiggen
(Faculty Advisor)

This project document meets the standards for appearance, conforms to the style and format requirements of the Computer Science Department of the University of North Dakota, and is hereby approved.

Graduate Director

7-18-02
Date

PERMISSION

Title: Data Collection with PDAs: A Computing Laboratory Application

Department: Department of Computer Science

Degree: Master of Science

In presenting this report in partial fulfillment of the requirements for a graduate degree from the University of North Dakota, I agree that Department of Computer Science shall make it freely available for inspection. I further agree that permission for extensive copying for scholarly purposes may be granted by the professor who supervised my work or, in his absence, by the Chairperson of the Department.

Signature 

Date 7/16/02

TABLE OF CONTENTS

LIST OF FIGURES ……...……...vii

LIST OF TABLES..……..viii

ACKNOWLEDGMENTS…….ix

ABSTRACT…….……...…….....…..……x

CHAPTER

I. INTRODUCTION..…….....……1

1.1 Problem Statement…….…..……1

1.2 Requirements……..……1

1.3 Structure of This Report…….……2

II. TOOLS AND DVELOPMENT ENVIRONMENT…….3

2.1 The Palm OS Development Environment……..…….3

2.2 Code Warrior IDE……..…….7

2.3 Palm Pilot Emulator…….……...9

2.4 Conduit Development Kit 4.02……..…11

2.5 Palm Application Process Flow……....12

III. SOFTWARE ENGINEERING TECHNIQUE AND PRINCIPLE…….…...15

3.1 Project Process.…….……15

3.2 System Engineering Models …...…….……16

3.2.1 Data Model……..…17

3.2.2 Data Flow Diagram…….…...18

IV. SPECIFICATIONS, IMPLEMENTATIONS AND TESTING……..…20

4.1 Specifications……..….20

4.1.1 The Palm Program…….…….20

4.1.2 The Conduit…….....23

4.2 Implementation of The Palm Program……..…...24

4.2.1 Interface design……..…....24

4.2.2 Data Structure…….....24

4.2.3 Functionality Implementation…….28

Table – The main interface……...28

Two Unique Features……34

4.3 Implementation of the Conduit…….37

4.3.1 Conduit Entry Points ……..…39

4.3.2 The Excel File. …….…..40

4.3.3 Implementation of the IPDClientNotify Interface….……..…41

CNotify Class.…….…..42

CDtCategories Class……..…42

CScoreReord Class…….…..43

CDtData Class…….…44

CSync Class…….45

4.3.4 The Synchronization Process…….45

4.4 Implementation of the Conduit……46

4.4.1 Two Testing Methods ……...47

Unit Testing……..….47

Integrated Testing……..…47

4.4.2 Testing Procedures and Results.……..…48

List/Enter/Edit Student Information

and Scores Testing …….……...49

Menu/Auxiliary Functionality Testing.…….……51

Integrated Testing/Conduit Testing……..…….52

REFERENCES…….……55

APPENDIX A…….…..57

APPENDIX B…….…..65

LIST OF FIGURES

1. Palm Application Process Flow……..……14

2. Level 0 Data Flow Diagram……18

3. Level 1 Data Flow Diagram…...…….19

4. The ScoreList Palm OS Application.……..21

5. C API-based and COM-based Conduits…….38

6. Synchronization Process…….46

LIST OF TABLES

1. The ScoreList Table ……17

2. Conduit Entry Points Called by the HotSync Manager.……..…39

3. The Properties of the CScoreRecord Class……..43

ACKNOWLEDGMENTS

Thanks to my dear parents for their love and support of twenty-eight years.

Appreciation to Dr. Tom Wiggen, who gave me guide, advice and urge that is needed for this project.

I also would like to thank all the staff of Department of Computer Science for their care and kindness to me.

To My Husband, Jun

ABSTRACT

In this report, we describe the design and implementation of a computer application for recording laboratory scores and uploading them to desktop computers. This computer application consists of two programs. The first program is a Palm OS program running on handheld devices. Users can carry a small handheld device (like Palm Pilot), with this program installed, to collect laboratory scores for each student in the class. The second program is a conduit program running within the Palm OS HotSync Manger on a desktop computer. It is used to transfer information collected in the laboratory between the Palm OS program and the Microsoft Excel application during a HotSync synchronization process. An Excel spreadsheet will be generated or updated with that information. Users can further process that information using the powerful Excel spreadsheet.

CHAPTER I

INTRODUCTION

1.1 Problem Statement

Lab instructors normally record each student's score on some paper during each lab class. And then later they have to input all the scores one by one onto computers. It is very time-consuming and very easy to make mistakes. It is a disirable to have a software application to make it possible for lab instructors to do a "paperless" job so that they can easily record student progress and transfer those data to a spreadsheet or other suitable PC application. Personal Digital Assistant (PDA) is chosen to do the job since it is a popular handy tool that is very convenient to use and easy to carry.

For this independent study, we designed and implemented a computer application for recording laboratory scores on a PDA or a computer and synchronizing the data between a PDA and a computer.

1.2 Requirements

For most computer users, the user interface is synonymous with the program itself. If the interface is poor, the software will not be used. Thus, one aspect of our project design was the design of the application's user interface. Second aspect of the project design is the format of the collected data. Another aspect, of course, is the coding, debugging and testing of the software system itself.

- User Interface: We want the user interface to be easy to understand, and convenient to handle. It was designed after consultation with prospective users of the software.
- Format of the collected data: The format of the collected data was chosen to be compatible with a standard spreadsheet format, so that the data can be easily imported into these applications.
- Coding, debugging, and testing: The coding for Palm Pilot platform was done in the C++ programming language. Initial testing and debugging was done on Palm Pilot simulator. A Palm Pilot m505 PDA was used for final testing and field trials by potential users.

1.3 Structure of This Report

Now the problem and requirement has been decided. We will introduce the Tools used to develop the application and the development environment in Chapter II. In Chapter III some of the software engineering techniques and principle are applied to the software developing. Specifications, implementation and testing are described in detail in Chapter IV. Appendix A includes the user manual and all the code is stored in the CD-Rom attached as Appendix B.

CHAPTER II

TOOLS AND DVELOPMENT ENVIRONMENT

The whole software includes two programs. One is a palm program running on the handheld device that collects data (scores) for each student in the class, the other is a desktop program running on the desktop computer that synchronizes the data on palm with the Microsoft Excel application on desktop.

The palm program, called ScoreList, is coded in C++ in the Code Warrior for Palm Platform 7.0 IDE. The synchronization program, called ScoreList Conduit, is coded in Visual Basic in the Microsoft Visual Studio 6.0 IDE, with the used of components from the Palm Conduit Development Kit 4.02.

In the following, we will briefly introduce each development tool and environment. Then, to give you a big picture, the process flow through all palm software components will be shown.

2.1 The Palm OS Development Environment

Writing applications for handhelds, specifically Palm OS platform devices, is a bit different from writing desktop applications because the Palm OS platform device is designed differently than a desktop computer. Also, users simply interact with the device differently than they do desktop computers.

Here are the differences that affect the design of a Palm OS® application.

1. Screen Size. The Palm OS device's screen is only 160x160 pixels, so the amount of information that can be displayed at one time is limited. So user interface must be designed carefully with different priorities and goals than are used for large screens, to get balance between providing enough information and overcrowding the screen.
2. Response time expected. On a PC, users don't mind waiting a few seconds while an application loads because they plan to use the application for an extended amount of time. By contrast, the average Palm user uses a Palm application 15 to 20 times per day for much briefer periods of time, usually just a few seconds. Speed is therefore a critical design objective for hand-held organizers and is not limited to execution speed of the code. The total time needed to navigate, select, and execute commands can have a big impact on overall efficiency. (Besides the Palm OS does not provide a wait cursor.) To maximize performance, the user interface should minimize navigation between windows, opening of dialog boxes, and so on. The layout of application screens needs to be simple so that the user can pick up the product and use it effectively after a short time. It's especially helpful if the user interface of your application is consistent with other applications on the device so users work with familiar patterns.
3. PC connectivity. PC connectivity is an integral component of the Palm OS platform device. The device comes with a cradle that connects to a desktop PC and with software for the PC that provides "one-button" backup and synchronization of all data on the device with the user's PC. Many Palm OS applications have a corresponding application on the desktop. A conduit has to be

written to share data between the device's application and the desktop's application.

4. Input methods. Handheld users don't have a keyboard or mouse. Users enter data into the device using a pen. They can either write Graffiti® strokes or use the keyboard dialog provided on the device. While Graffiti strokes and the keyboard dialog are useful ways of entering data, they are not as convenient as using the full-sized desktop computer with its keyboard and mouse. Therefore, users should not be required to enter a lot of data on the device itself.
5. Power. The Palm OS platform device runs on batteries and thus does not have the same processing power as a desktop PC. It is intended as a satellite viewer for corresponding desktop applications. If your application needs to perform a computationally intensive task, you should implement that task in the desktop application instead of the device application.
6. Memory. The Palm OS device has limited heap space and storage space. Different versions of the device have between 512K and 8MB total of dynamic memory and storage available. The device does not have a disk drive or PCMCIA support. Because of the limited space and power, optimization is critical. Optimization for heap space, speed, code size makes the application fast and efficient.
7. File System. Because of the limited storage space, and to make synchronization with the desktop computer more efficient, Palm OS does not use a traditional file system. Data is stored in memory chunks called records, which are grouped into databases. A database is analogous to a file. The difference is that data is broken down into multiple records instead of being stored in one contiguous chunk.

Database is edited in place in memory instead of creating it in RAM and then writing it out to storage to save space.

8. Backward Compatibility. Different versions of the Palm OS platform device are available, and each runs a different version of the Palm OS. Users are not expected to upgrade their versions of the Palm OS as rapidly as they would an operating system on a desktop computer. Updates to the OS are designed in such a way that you can easily maintain backward compatibility with previous versions of the OS, so that the application is available to more users.

The Palm OS has a pre-emptive multitasking kernel. However, the User Interface Application Shell (UIAS), the part of the OS responsible for managing applications that display a user interface, runs only one application at a time. Normally, the only task running is the UIAS, which calls application code as a subroutine. The UIAS doesn't gain control again until the currently running application quits, at which point the UIAS immediately calls the next application as another subroutine.

When the system launches an application, it calls a function named **PilotMain** (similar to the **main** function in a C program) and passes it a launch code. The launch code may tell the application to start and display its user interface, in which case it will start up its event loop and process the event queue.

A Palm OS application is event-driven, receiving events from the OS and either handling them or passing them back to be handled by the OS itself. An event structure describes the type of event that has taken place (for example, a stylus tap on an on-screen button), as well as information related to that event, such as the screen coordinates of a stylus tap. During a normal launch, execution passes to the application's *event loop*,

which retrieves events from the event queue and dispatches them according to the type of event.

Memory in the Palm OS is divided into dynamic and storage areas, each with its own unique limitations. The entire dynamic area of the device's RAM is used to implement the dynamicheap. A heap is a contiguous area of memory that manages and contains smaller units of memory. These smaller units are called chunks. A chunk is a contiguous area of memory between 1 byte and slightly less than 64KB in size. All data in the Palm OS environment are stored in chunks.

The dynamic heap provides memory for several purposes:

- Application and system global variables
- Dynamic allocations by the system, such as the TCP/IP and IrDA stacks
- Stack space for the running application
- Temporary memory allocations
- Dynamic allocations by applications

Any memory on the device that is not dedicated to the dynamic heap is divided into a number of storage heaps. The Palm OS power management scheme works in such a way that the device is never really "off," only resting.

2.2 Code Warrior IDE

CodeWarrior Integrated Development Environment (IDE) is used to develop software for various operating systems using programming languages such as C, C++, and the Java programming language. It is an application that provides a set of tools for developing software using a graphical user interface (GUI). IDE can be used to perform

code editing, navigating through code, examining code, compiling, and linking code. You can also configure options for code generation, project navigation, and other operations.

CodeWarrior products include various IDE tools. The tools that ship with most IDE versions are: a source-code editor, a source-code browser, compilers, linkers, assemblers, and a debugger.

The CodeWarrior IDE provides flexibility in several ways. First, IDE can be run on a variety of host computers while you develop software. The CodeWarrior product that you purchased determines the hosts available to you. Second, The CodeWarrior product that you purchased determines the platform targets for which you can create code. For example, you can use the IDE to develop software for Pentium computers running the Microsoft Windows operating system. Besides, IDE can be used to develop a program, plug-in, library, or other executable code using different programming languages such as C, C++, and Java. Last, the IDE uses plug-ins to support several tools, including the compilers, linkers, and the CodeWarrior debugger. The IDE may also include additional plug-ins, depending on the version that you purchased. You can also use third-party plug-in tools, such as plug-in compilers and debuggers.

The CodeWarrior IDE uses Project, a collection of related files and configuration settings to organize software development. Using projects avoids writing complicated build scripts (makefiles). Instead of editing build scripts, you can use simple mouse and keyboard operations to manipulate CodeWarrior projects. You can use a single project to create and manage several configurations of your software for use on various computer platforms.

Additionally, the Code Warrior for Palm OS platform includes a constructor. Constructor is a visual resource editor used to specify how your application appears to the end user. Constructor is used to build a visual interface for Palm OS software applications, used by devices like Palm connected organizers. Using Constructor, you create and edit several types of resources for Palm OS software applications.

Constructor manages these types of resources for Palm OS software: Application Resources, Forms and Controls, Menus, Character Strings, String Lists, Category Names, Alerts, and Icons and Bitmaps.

Constructor creates a C header file containing preprocessor definitions for the resources defined in your Constructor for Palm OS project file. You can include this header file in the C source code files in your Palm OS application's project.

2.3. Palm Pilot Emulator

The Palm OS Emulator is a hardware emulator program for the Palm Computing platform, which means that it emulates the Palm hardware in software, providing the ability to test and debug Palm OS software on a Macintosh, Unix, or Windows-based desktop computer.

When a Palm OS application is run with the Palm OS Emulator on your desktop computer, the Palm OS Emulator fetches instructions, updates the handheld screen display, works with special registers, and handles interrupts in exactly the same manner as does the processor inside of Palm Computing platform handhelds. The difference is that the Palm OS Emulator executes these instructions in software on your desktop computer. The Palm OS Emulator displays an on-screen image that looks exactly like a

Palm™ connected organizer. Many type of Palm handheld devices are provided online. Mouse can be used on the desktop computer just as the stylus is used on a Palm connected organizer.

Palm OS Emulator accurately emulates Palm Computing platform hardware devices, and includes the following features:

- An exact replica of the Palm device display, including the silkscreen and Graffiti areas
- Emulation of the Palm stylus with the desktop computer pointing device
- Emulation of the Palm device hardware buttons, including:
    - Power on/off button
    - Application buttons
    - Up and down buttons
    - Reset button
    - HotSync button
- Ability to zoom the display for enhanced readability and presentation
- Screen backlighting
- Communications port emulation for modem communications and synchronizing

Palm OS Emulator also provides the following capabilities that extend the standard Palm device interface.

- Ability to enter text with the desktop computer
- Configurable memory card size, up to 8MB

Besides, Palm OS Emulator provides a large number of debugging features that help to detect coding problems and unsafe application operations. Palm OS Emulator includes the following debugging features and capabilities:

- Ability to use an automated test facility called Gremlins, which repeatedly generates random events
- Support for external debuggers, including Palm Debugger, the Metrowerks CodeWarrior debugger, and GDB
- Monitoring of application actions, including various memory access and memory block activities
- Logging of application activities, including events handled, functions called, and CPU opcodes executed by the application
- Profiling of application performance

2.4. Conduit Development Kit 4.02

A conduit is a plug-in to the HotSync® technology that runs when you press the HotSync button. A conduit synchronizes data between the application on the desktop and the application on the hand-held device. The Conduit SDK is for you to write a conduit.

The Conduit Development Kit (CDK) 4.02a allows developers to create conduits using Microsoft Visual C++ 6.x, Visual Basic 6.0, and WebGain VisualCafe 4.0/4.0a.

The Conduit Development Kit (CDK) for Windows includes several indispensable utilities to help developers create and debug conduits for Windows. Utilities include but are not limited to the following ones:

Conduit Configuration Utility: A developer-only application that allows you to register your conduit with HotSync® Manager, edit or delete conduit information, and modify some HotSync Manager settings

HotSync Manager: A user application that manages synchronization of data between the handheld and the desktop. The CDK includes versions of HotSync Manager that shipped with Palm powered™ handhelds and a specialversion provided only in the CDK to use with the Conduit Inspector.

Install Tool Utility: User application that installs Palm OS® applications and data on a handheld. The CDK includes a version of Install Tool for each version of HotSync Manager.

Conduit Inspector Utility: A developer-only application that logs detailed status information from HotSync Manager in real time to help you debug your conduit.

User Info Utility: A developer-only application that adds, deletes, and modifies desktop user information.

2.5 Palm Application Process Flow

A complete palm application needs seven software components to make it work. They are:

- Desktop applications, which are developed by you or another developer, run on the desktop computer and operate on data that is sent to and/or retrieved from the handheld. In our case, the desktop application is the Microsoft Excel application.

- Palm OS applications developed to run on any Palm OS handheld. These are also referred to as a handheld applications. In our case, it is the ScoreList palm OS application.
- The HotSync Manager, supplied by Palm, runs on the desktop computer and communicates with the handheld. The HotSync Manager awakens when the user selects HotSync on the Palm HotSync Manager desktop application, and the HotSync Manager calls each of the conduits that are properly installed and configured on the user's desktop computer.
- The HotSync client launches on the handheld when the user presses the HotSync button on the cradle. This handheld application wakes up the HotSync Manager and responds to Sync Manager requests to access databases on the handheld. It is an integral part of the Palm OS.
- Conduit*s* are the programs that the HotSync Manager runs to interact with specific data on the handheld. You use the Conduit Development Kit to create conduits. Generally, when a handheld application is written, a conduit is written to synchronize its data with data on the desktop. In our case, it is the ScoreList conduit.
- The Sync Manager API provides a programmatic interface that conduits use for communicating with a handheld. This communications API allows the conduit to remain independent of the connection type between the handheld and the desktop computer. It is supplied by Palm.
- Notifiers are called by the HotSync Manager to tell programs on the desktop computer that the HotSync Manager is running. A notifier is a Windows DLL that

you create to tell your desktop application that your conduit is modifying its desktop data. This behavior ensures that the application and its associated conduit are not both changing data at the same time. Note that notifiers are called only by the Windows version of the HotSync Manager.

The following figure shows the process flow through these components.

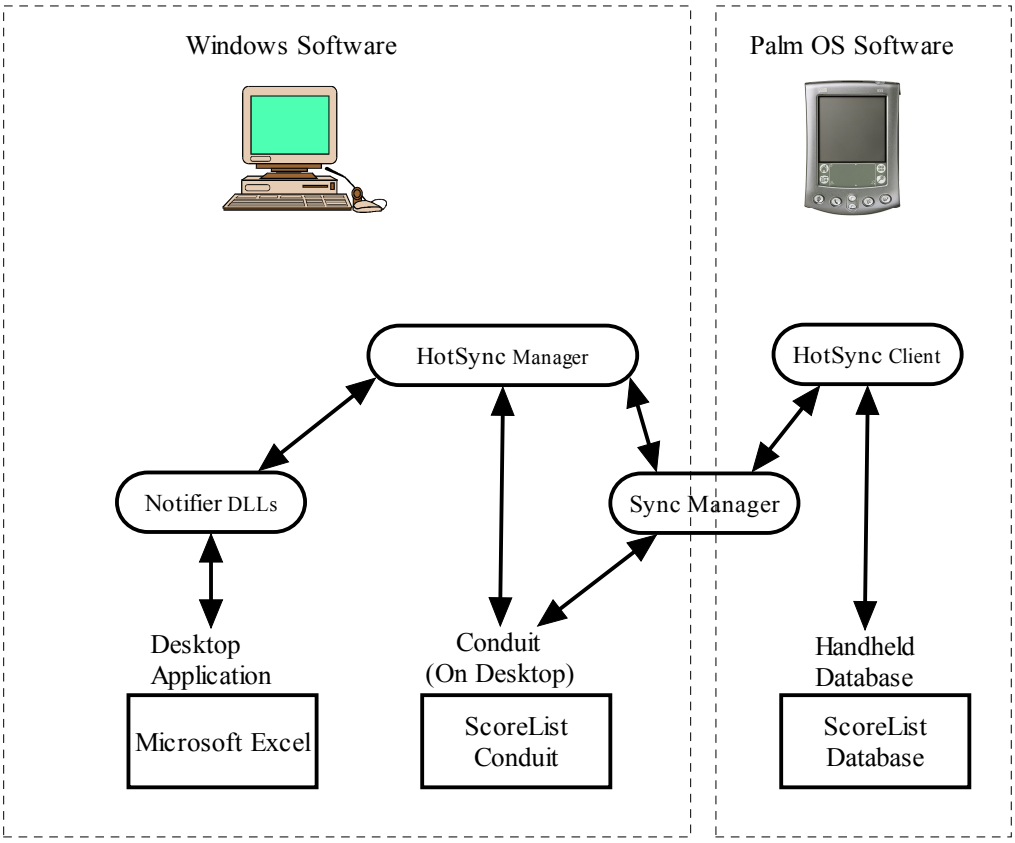


Figure 1. Palm Application Process Flow

CHAPTER III

SOFTWARE ENGINEERING TECHNIQUE AND PRINCIPLE

During the development of the project, some software techniques and principles are applied.

3.1 Project Process

The incremental process was used for this project. The incremental model, can apply linear sequences repetitively and in a staggered fashion as calendar time progresses. It combines the iterative philosophy of the prototyping model as well. After each increment is done, a deliverable increment of the software is generated.

The first increment is to build the main interface, the list form and the edit form, which is used to add or edit a record. The user should be able to add or edit a record, show records (with NAID, Last Name, First Name and Scores and Note) correctly, to show records in different format, change a record's category and also should be able to create note.

The second increment is to add the functionalities of entering scores and showing column titles. First, the user should be able to enter a score rapidly. Second, since the user may want to enter any number of scores, we do not want to set up a fixed number of score columns so score column should be created dynamically. Third, the palm screen is not able to show a lot of score columns at the same time so we need to be able to show

score columns by scrolling left and right. Lastly, we need to show the column titles so the user knows what they are viewing clearly.

The third increment is to add some additional functionalities, like menus, cut and past functionality, record duplication, showing an about form, etc. These are not core functionalities but it is nice to have.

The fourth increment will be the synchronization function. The users should be able synchronize the data between desktop and Palm handhold with different ways like “overwrite handheld”, “overwrite desktop”, “synchronize”. A conduit should be built for this purpose.

Each increment includes analysis, design, code and test.

The advantages of the incremental model are overwhelming. It is appropriate for the situation when only one person is involved in coding, debugging, and most testing. So increments can be implemented step by step. In addition, according to the result of use and/or evaluation of the previous increment, a plan, which will address the modification of the previous increment to better meet the requirement and the delivery of additional features and functionality, will be developed for the next increment.

3.2 System Engineering Models

Software engineering occurs as a consequence of a process called system engineering. System engineering focuses on a variety of elements, analyzing, designing, and organizing those elements into a system that can be a product, a service, or a technology for the transformation of information or control. System engineering is a modeling process. During the process of engineering, we create several models to

describe the date structure, the function and system behavior. In this document, we developed a data model and a data flow diagram to define the main function and behavior of our software. We will discuss them one by one in the following.

3.2.1 Data Model

Data design (sometimes referred to as data architecting) creates a model of data and or/ information that is represented at a high level of abstraction. This data model is then refined into progressively more implementation—specific representations that can be processed by the computer—based systems. The structure of data has always been an important part of software design. At the program component level, the design of data structures and the associated algorithms required to manipulate them is essential to the creation of high—quality application.

I managed all the data in one table (shown in Table 3.1), one relational database, which I will explain in more detail in Chapter 4. The table's name is ScoreList. Different with normal tables, this table has a dynamic number of score columns.

Table 1. The ScoreList Table

NAID	Last Name	First Name	(Dynamically Created) Lab1	(Dynamically created) Lab2	(Dynamically Created) Lab3	(Dynamically Created) ……	Note
String	String	String	String	String	String	String	String

3.2.2. Data Flow Diagram

A data flow diagram (DFD) is a graphical representation that depicts information flow and the transforms that are applied as data move from input to output. DFD may partition into levels that represent increasing information flow and functional detail. For this system we are developing, two-level DFD is used.

Level 0 data flow diagram is also called context model or a fundamental system model. It represents the entire software element as a single bubble with input and output data indicated by incoming and outgoing arrows respectively. In addition, the information flow continuity and the balancing between input and output data have to be maintained.

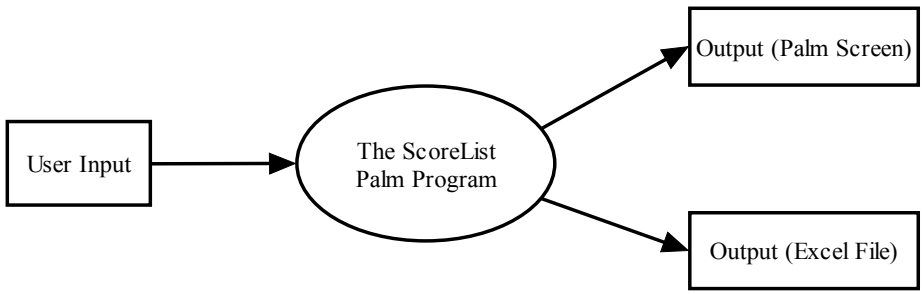


Figure 2. Level 0 Data Flow Diagram

There are two kinds of input data, one is student information data, including any part of the records, the other one is the user command such as “delete”, “add”, “edit”.

There is one kind of output data from the online recipe software. It shows the mostly updated score list.

Since there is only one bubbles in the level 0 DFD, it is unclear for the algorithm this is applied to transform the input and the output that is produced. As a result, the restrictions and limitations imposed on the process are ambiguous. We can partition the

level 0 DFD to level 1 DFD to reveal more detail. There are 5 processes in this Data Flow Diagram,

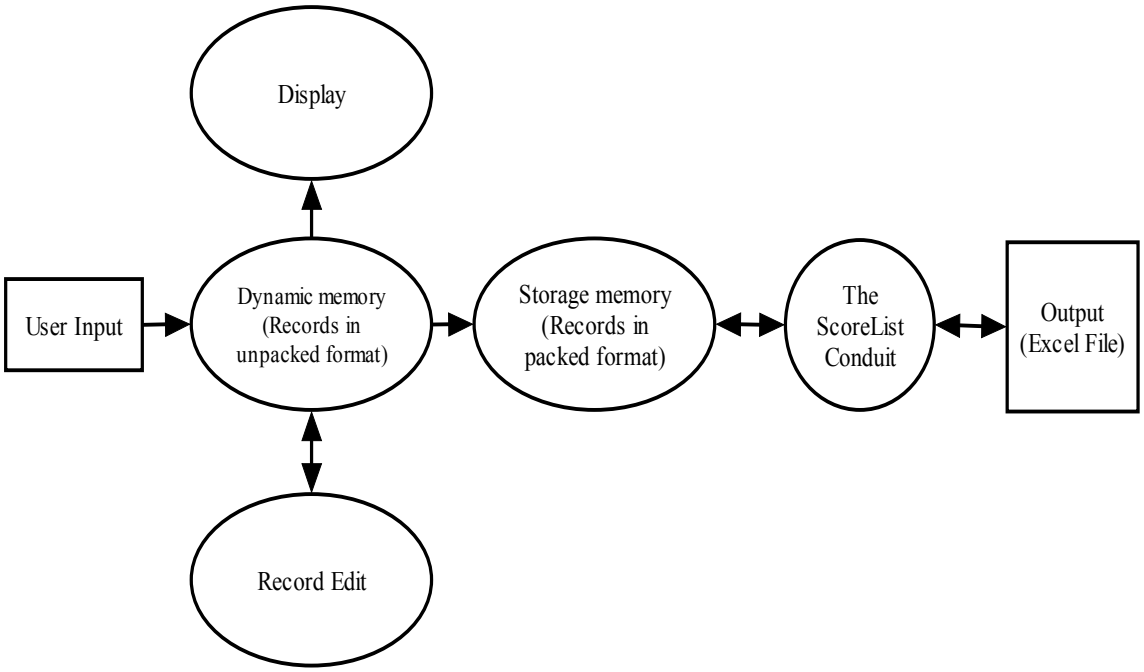


Figure 3. Level 1 Data Flow Diagram

The data model and the data flow diagram provide use some meaningful insight into software requirements. It will help use in implementing the software system in Chapter IV.

CHAPTER IV

SPECIFICATIONS, IMPLEMENTATIONS AND TESTING

In this chapter, the specifications of the software will be presented. Then we will explain how the software is implemented. Some important features will be discussed in details. Last, software testing is done to make sure its quality.

4.1 Specifications

The specifications for the palm program and the conduit will be presented separately in the following.

4.1.1 The Palm Program

The program uses a spreadsheet like interface. Each record (row) can have NAID (student id), name (last name and first name), and then up to 255 numbers of scores and a note to record anything about the student. Records can be grouped into different categories, which are customizable. For example, a class may have 3 sections, Section A, Section B and Section C. You can assign each student into its appropriate sections. Figure 4 is what it looks like.

It has the following features:

- Running at Palm 2.0 or higher
- Scores are 3 levels, 0 1 or 2. “?” means no score.

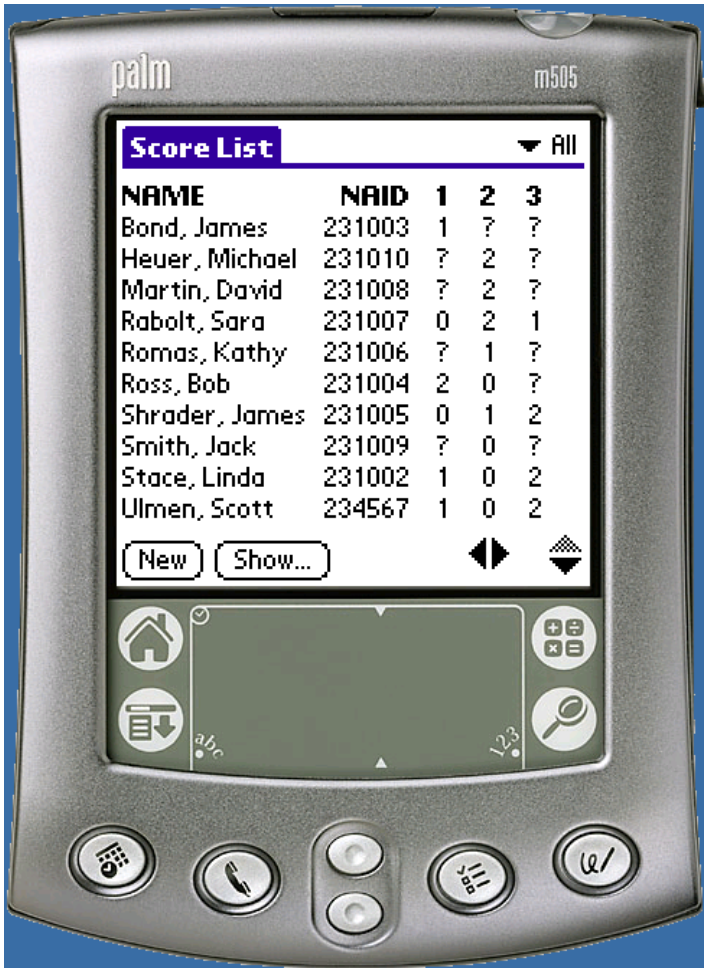


Figure 4. The ScoreList Palm OS Application

- Divide students into different categories. Like if a class has three sections, like section A, B and C. You can create categories Section A, Section B and Section C.
- Record scores for as many students as you want as long as you have enough memory on your palm.
- You can show the records in four ways:
    - Name and NAID (sorted by name)
    - NAID and name (sorted by NAID)
    - NAID only (sorted by NAID)

  - Name only (sorted by name)

- You can enter up to 255 scores for each student
- There are four scroll buttons, Scroll Left, Scroll Right, Scroll Up and Scroll Down, for you to browse records and scores.
- You can enter a note for each record/student. Note is a good place for you to enter something about the student and his/her scores.
- Another feature about this program is to show column name for each column. This feature is not often seen in standard palm programs.
- It has menus and shortcuts that allow you to do things like cut, copy, paste strings, duplicate records, etc. Look back to Figure 4 to see the program interface.
- Limitation: You can record scores for only one class.

4.1.2 The Conduit

The conduit program is to synchronize data between palm and desktop. The first time you synchronize, it will generate a file called scorelist.xls on your palm user directory. It is a standard Microsoft Excel file. It contains all records from palm. Whenever you make changes, like changing score, adding score or deleting score, adding student, editing student information, deleting student on palm or desktop, all changes will be available in both palm and the excel file after synchronization when you choose a correct HotSync action.

There are four kinds of HotSync actions:

- Synchronzing the files
- Handheld overwrites Desktop

- Desktop overwrites handheld
- Do nothing

It is very clear that what will happen for the last three kinds of HotSync action. For the first kind, it is a little complicated. Here is how the synchronization is done:

This HotSync action can keep all changes made either on Palm or Desktop. In the case of conflicting changes on the same record, usually the record on palm will overwrite the corresponding record on desktop. So changes made on desktop will be lost after synchronization. However, there is one exception: If the number of scores of the record on desktop is larger than that of the record on palm, the desktop record will overwrite the palm record.

Depending on what you need, you should choose different HotSync actions. Here are the instructions:

- To keep all changes you made in either Palm or Desktop available to both places, choose Synchronize the files. But you need to know when the same record is changed in both places, the palm record overwrites the desktop one with one exception: when the number of scores in the Desktop record is larger than that in the Palm record, the desktop record overwrites the one on Palm.
- If you only want to keep the records on Palm, choose Handheld overwrites Desktop.
- If you only want to keep the records on Desktop, choose Desktop overwrites Palm
- If you don’t want to synchronize ScoreList database in the next synchronization, choose Do Nothing.

The excel file ScoreList.xls contains section (category), NAID, name and scores. The note is stored in the comment field of column NAID.

You can create categories on the fly. If you enter a new section in the Section column, it will create a category on palm with the same name automatically.

4.2 Implementation of The Palm Program

Implementations include interface design, data structure and functionality implementation.

4.2.1. Interface design

The interface design goals are:

- Rapid input of lab scores (Entering a score just needs two taps)
- Simple, intuitive
- Consistent with other standard palm programs so a user will learn it in no time

Figure 4 shows the main interface of the palm program. As you can see, it is very simple and intuitive. You should be able to figure out how to use it in no time.

4.2.2. Data Structure

Externally, each record is shown in 5 columns, title (a combination of NAID, last name and first name), score 1, score 2, score 3 and note. Internally, each record also contains five fields but they are defined differently. Internally each record is represented in two different ways, unpacked and packed. The unpacked representation is convenient

to get and set each field. But, to save storage space, the packed representation is used when records are saved. The following data structure is the unpacked representation.

```
// LibDBRecordType
// This is the unpacked record form used by the application.  The fields
// array contains pointers to each of the text fields in the record.
// Each pointer in the array is either NULL if the field is empty, or
// it points to the contents of the field.
typedef struct {
        char        *fields[libFieldsCount];
} LibDBRecordType;
```

Here are the fields defined in a record:

```
// LibFields
// Enum for identifying the text fields in a book record.
typedef enum {
        libFieldNaid = 0,
        libFieldLastName,
        libFieldFirstName,
        libFieldScores,
        libFieldNote,
        libFieldsCount
} LibFields;
```

As you see, the title column shown externally is actually represented in three fields internally, that is libFieldNaid, libFieldLastName and libFieldFirstName. The reason is that when editing a student, we need to show these three pieces of information

separately thus we need to store these information separately, at the same time, these information can be easily combined together in one column on the form.

The three score columns shown externally is actually represented in one field, libFieldScores, internally. Each score column shown externally is actually just one character in the libFieldScores field. The first character in the libFieldScores field will be shown in the first score column, the second character is in the second score column, … , etc.. Since each text field can only have up to 255 characters, we can only have up to 255 scores for each record. Since each score is represented by one character, we can show up to 255 levels of scores.

Since for any field, the length could vary for each record, if we allocate a fixed length of storage space for each field, we could waste a lot of space. In order to save space, each record is stored in a packed format represented by the following data structure.

```
// LibPackedDBRecord
// This is the packed record form stored in the application's database.
// Each field is only stored if it contains any data.  The
// LibDBRecordFlags structure indicates which fields exist for the
// record.  The firstField member is actually the location of the first
// character of the first text field that contains data.  Each string
// representing a text field's contents is null-terminated.
// The various offset variables are necessary to provide direct
// in-place access to key fields when sorting records.  The exception to
// this is noteOffset, which is necessary to find the note field for
// in-place editing.
```

```
typedef struct {
        LibDBRecordFlags  flags;
        unsigned char     naidOffset;
        unsigned char     lastNameOffset;
        unsigned char     firstNameOffset;
        unsigned char     scoresOffset;
        unsigned char     noteOffset;
        char              firstField;
} LibPackedDBRecord;


// LibDBRecordFlags
// Since the packed record format in LibPackedDBRecord does not store
// empty fields, Scorekeeper uses LibDBRecordFlags to keep track of which
// fields contain data.  0 = empty field, 1 = a field containing text.
typedef union {
        struct {
   unsigned reserved    :11; // Extra space for possible future fields
            unsigned note          :1;  // Set if record contains a note handle
            unsigned scores        :1;  // Set if record contains scores
            unsigned firstName  :1;  // Set if record contains an student's first name
            unsigned lastName   :1;  // Set if record contains an student's last name
            unsigned naid     :1;  // Set if record contains a student's naid (student id)
                } bits;
        UInt16 allBits;
} LibDBRecordFlags;
```

When retrieving a record from memory, it needs to be converted from the packed format to unpacked format. When saving a record to memory, it needs to be converted from the unpacked format to packed format.

4.2.3. Functionality Implementation

Table – The main interface

The main interface, which lists all records and allows you to enter scores, is mainly a table. Tables are some of the most complex user interface elements in the Palm OS and, hence, some of the most difficult to implement. Fortunately, many internal functions of tables are handled by the Palm OS table manager, and much of the code required to operate a table can be used with slight modification by different palm applications. So learning one table example application will let you quickly master table programming. Basically what you need to create a new table application is to modify some functions slightly and reuse them in your new application.

Palm OS tables are essentially containers for a variety of other form elements, such as pop-up lists, text fields, and check boxes, and as such, each table maintains a complex array of subordinate controls. Each control is an item that corresponds to a column. There are several item types including checkBoxTableItem, customTableItem, dateTableItem, labelTableItem, numericTableItem, popupTriggerTableItem, textTableItem, textWithNoteTableItem and narrowTextTableItem. You can see the Palm OS companion [4] for more information on these item types. Basically customTableItem allows the most flexibility in determining what to be displayed in a column. The ScoreList table uses customTableItem type for all its five columns.

There are two basic tasks that you need in order to operate a table: Initializing a table and handling table events. We will briefly take a look at how these tasks are implemented in the ScoreList program.

1. Initializing a Table

Before we can implement user interaction with a table, or even use the table to display data, we must prepare the table for use. Initializing a table primarily involves telling the table manager what item type each cell of the table should be, along with setting up callback functions for certain columns to perform custom drawing routines, or to save and retrieve text from fields in the table.

We need to initialize a table before the system draws it to the screen. In the ScoreList program, the ListFormInit function, which initializes the form as well as the table, is placed in the ListFormHandleEvent function, the main form's event handler.

First, it uses an internal palm function TblSetItemStyle to set item type for each column in function ListFormInit, which initializes the controls including the table on the list form:

```
//set item type for each column each row
TblSetItemStyle(table, row, titleColumn, customTableItem);  //note: title column will display naid and/or name depending on the setting
TblSetItemStyle(table, row, score1Column, customTableItem);
TblSetItemStyle(table, row, score2Column, customTableItem);
TblSetItemStyle(table, row, score3Column, customTableItem);
TblSetItemStyle(table, row, noteColumn, customTableItem);
```

After setting other attributes for each columns, like font, usability, masks (PalmOS 3.5 or later), etc, the program needs to set one or more callback routines to draw each column:

```
// Set the callback routine that will draw the records.
TblSetCustomDrawProcedure(table, titleColumn, ListFormDrawRecord);
TblSetCustomDrawProcedure(table, score1Column, ListFormDrawRecord);
TblSetCustomDrawProcedure(table, score2Column, ListFormDrawRecord);
TblSetCustomDrawProcedure(table, score3Column, ListFormDrawRecord);
TblSetCustomDrawProcedure(table, noteColumn, ListFormDrawRecord);
```

The ListFormDrawRecord function is a callback function that will draw each column when displayed.

Then, we need to load data (records) into the table to be displayed. This is implemented in the function ListFormLoadTable, which we will explain in more details later.

2. Handling table events

Handling table events is done together with other controls' events on the main form (list form) in function ListFormHandleEvent. The following events are handled in this function:

- frmOpenEvent: When this event is triggered, list form is initialized and drawn.
- frmTblSelectEvent: when you select a record on the list form, this event will be triggered. Depending which column you selected, different actions will be performed. If you select the title column, the Edit Form will be displayed for you

to edit the student's NAID, last name and first name. If you select one of the score column, a list form with three categories, 0, 1 and 2, will be displayed. You can pick one from the list to enter a score.

- ctlSelectEvent: There are two buttons on the list form, the New command button and the Show button. There is also a List control, Category. When you tap on the New button, ctlSelectEvent will be fired, function EditFormNewRecord will be called to show the EditForm to create a new record. When you tap on the Show button, ctlSelectEvent will be fired, function FrmPopupForm will be called to show the ScoreList Preference form for you to select how to display record. When you select the pop up category list on the right upper corner on the form, a list form with three categories will be displayed. You can pick one from the list to display records that only belong to a specific category or all records.
- ctlRepeatingEvent: There are four repeating buttons. ListScrollUpRepeating, ListScrollDownRepeating, ListScrollLeftRepeating, ListScrollRightRepeating. When you tap on these buttons, corresponding action will be taken. Ex. When you tap on ListScrollDownRepeating button, the list will scroll down one page and more records (if there are any) will be shown. When you tap on ListScrollRightRepeating button, next three score columns will be dynamically created and shown, which will be explained in more details later.

3. Loading data into table

After initializing the table, we need to load data into table. Loading data into table may be seen as a part of initializing table. It is implemented in function ListFormLoadTable. Because ListFormLoadTable completely redraws the items

displayed in the table, the highlighted selection becomes invalid. The first thing ListFormLoadTable does is to retrieve a handle to the List form's table and unhighlight the table's current selection with the TblUnhighlightSelection function. The ListFormLoadTable function then retrieves the number of rows the table can display at once using the helper function ListFormNumberOfRows. In the ScoreList program, the font size for the list form is fixed so the number of rows is a constant number, 11.

Once the number of visible rows is stored in the variable visibleRows, ListFormLoadTable looks for an appropriate record in the database to use as a starting point for filling in the table. The global variable gTopVisibleRecord stores the record ID of the record at the top of the table. If the program is just starting up, gTopVisible Record is equal to 0, representing the first record in Librarian's database. In order for a record to be a good candidate for gTopVisibleRecord, the following things must be true:

(1) There must be enough displayable records after gTopVisibleRecord in Librarian's database to fill the entire table, leaving no blank rows at the end.

(2) Failing that, gTopVisibleRecord should be the first displayable record in the database.

The ListFormLoadTable function first determines the viability of gTopVisibleRecord, which is the starting point in the database, it can then get to the business of actually filling in the table.

For each visible row, ListFormLoadTable searches for the next displayable record using SeekRecord; if it doesn't find one, ListFormLoadTable is finished filling in records and breaks out of the for loop. The ListFormLoadTable function sets each row that does have a record usable with TblSetRowUsable, and then marks that row invalid with

TblMarkRowInvalid so that the table manager redraws the row when it next draws the table. Most important, the for loop stores the index of the record as the row's ID number with the TblSetRowID function. Every row in a table has an ID value, which you may use to store whatever unsigned 16-bit integer value you wish. This ID number is an ideal place to stash the database index of the record displayed in a particular row.

The Palm OS also provides the companion functions TblGetRowID and TblFindRowID. The TblGetRowID function simply returns the ID value assigned to a row, given the row number. If you need to retrieve the row number and all you have is the row's ID number, use the TblFindRowID function instead.

After filling in the table, ListFormLoadTable hides with a while loop the rows that do not contain any data:

```
// Hide the items that don't have any data.
numRows = TblGetNumberOfRows (table);
while (row < numRows) {
TblSetRowUsable(table, row, false);
row++;
}
```

Finally, ListFormLoadTable calls Librarian's ListFormUpdateScrollButtons function, which makes sure that any changes to the table made by ListFormLoadTable are reflected in the appearance the repeating arrow buttons in the lower right corner of the List form.

Two Unique Features

The ScoreList palm application has two unique features that are not often seen in other palm applications. They are listed in the following.

1. Show Column Title

Palm applications seldom display each column's title. The internal Address List application, ToDo application and the Memo application all do not show the column titles. I think the reason behind this is that it is too obvious to be displayed. However, for our ScoreList application, we feel the need to show the column title in order for users to see clearly what they are viewing.

Since this is not a common programming, there is no code we can refer to. We did some trick in order to show the column title.

As we know, the visible number of rows (called visibleRows) for the ScoreList table is 11. Since we need to show the column title, we can only show 10 actual rows/records. Here are the changes we made in order to get this work.

- In function ListFormLoadTable, when determining gTopVisibleRecord, we should remember our actual visible rows (or more exactly records) is visibleRows –1, not visibleRows. If you have 15 records in total, currently you are showing the first 10 records (gTopVisibleRecord is zero), then when showing next page, gTopVisibleRecord should be set to 15-10=5 (zero-based), not 15-11=4.
- In function ListFormDrawRecord, we need to draw different for the first row. The first row should display the column titles, not the actual records.

- When handling the ctlSelectEvent, if the first row is selected, it should be handled differently from other rows. In fact, when the first row is selected, all handle code should be disabled.

2. Dynamically Create Score Columns

The score columns are dynamically created as needed. Initially, only three score columns are created. A screen can show three score columns at a time, which means you are able to see three scores for a student at a time. When you want to create and/or see more scores, you can press the Scroll Right button. When you want to previous scores, you press the Scroll Left button.

To implement this feature, we use a global gCurrentScorePosition to indicate the position of the first score column currently displayed. If what currently displayed is the first three scores for the record, gCurrentScorePosition is equal to zero; if what currently displayed is the 4^{th} to 6^{th} scores, gCurrentScorePosition is equal to 3. Since we display three scores each time, gCurrentScorePosition will always be equal to 0, 3, 6, …

When the user taps the ScrollLeft or ScrollRight button, gCurrentScorePosition is adjusted, then the table is reloaded and redrawed. This is implemented in function ListFormHandleEvent:

```
        ….
case ctlRepeatEvent:
    witch (event->data.ctlEnter.controlID) {
        case ListScrollUpRepeating:
            ListFormScroll(winUp, 1, false);
            // Leave unhandled so button can repeat.
```

```
                reak;
case ListScrollDownRepeating:
                ListFormScroll(winDown, 1, false);
                // Leave unhandled so button can repeat.
                break;
case ListScrollLeftRepeating:
                if(gCurrentScorePosition >= gNumScoreColumnsPerPage)
                gCurrentScorePosition = gCurrentScorePosition - gNumScoreColumnsPerPage;
                ListFormLoadTable();
                table = GetObjectPtr(ListTable);
                TblRedrawTable(table);
                handled = true;
                break;
case ListScrollRightRepeating:
                if(gCurrentScorePosition <= (255 - gNumScoreColumnsPerPage))
                gCurrentScorePosition = gCurrentScorePosition + gNumScoreColumnsPerPage;
                if(gMaxScoreColumns < gCurrentScorePosition +
gNumScoreColumnsPerPage)
                gMaxScoreColumns = gCurrentScorePosition + gNumScoreColumnsPerPage;
                ListFormLoadTable();
                table = GetObjectPtr(ListTable);
                TblRedrawTable(table);
                handled = true;
                break;
                default:
                break;
                }
                break;
```

…

When displaying scores, we have to consider what score column current is at. In function ListFormDrawRecord, when drawing scores, function GetGrade is used to get the score based on the record, gCurrentScorePosition and the column number.

4.3 Implementation of the Conduit

The ScoreList Conduit is built using the COM Sync Suit from the Palm Conduit Development Kit 4.02.

The COM Sync Suite is a component of the Conduit Development Kit (CDK) for Windows from Palm, Inc. It provides COM objects/methods/properties, the IPDClientNotify interface, COM Sync software layer, documents, utilities, and the COM Sync Installer to help developers create COM-based conduits that run on Windows computers. Key to the success of the Palm OS platform, conduits are software objects that exchange and synchronize data between an application running on a desktop computer and a Palm Powered handheld.

The COM Sync Suit allows you to directly access information on Palm Powered handhelds from a desktop computer using any COM-enabled programming language such as Visual Basic (VB), Borland C/C++, Java, and Visual C/C++.

The ScoreList conduit is implemented as an Active X DLL using Visual Basic. Visual Basic is used instead of Visual C++ or Java for the following reasons:

- Visual Basic is a rapid application development tools, it is faster to develop an ActiveX DLL than Visual C++

- It is more convenient to use Visual Basic to interact with the Excel application.

The COM Sync module consists of DLLs that act as client and server during the HotSync process. The server is a single DLL and a standard HotSync conduit. The server DLL "wraps" the Sync Manager API with a COM interface, while the client exposes that COM interface using an object model. The client DLLs supply you with an object model that communicates with the server. Figure 5 compares how standard C API-based conduits and COM-based conduits communicate with the handheld.

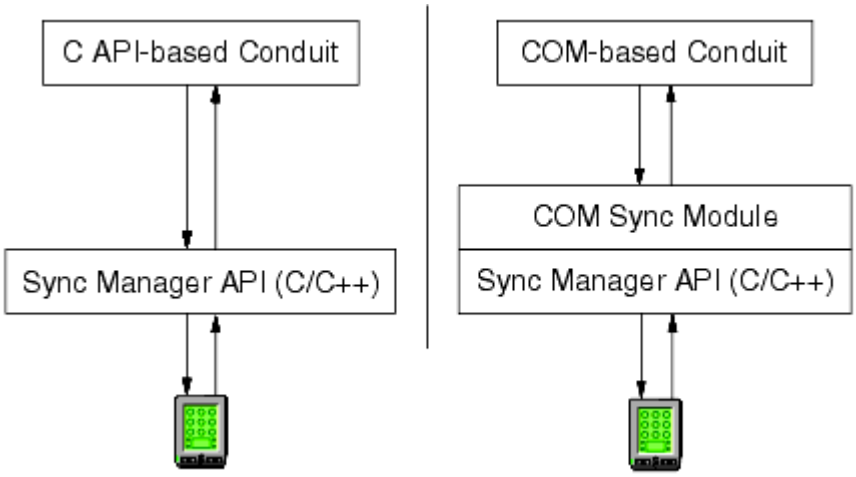


Figure 5. C API-based and COM-based Conduits

The COM Sync module extends the Sync Manager API functionality. Your conduit can open and access as many databases as you wish. The COM Sync module adds a "ReadNext" to the current set for convenience. Each database object has a categories object permitting easy category management. There is a utility object that permits easy insertion and extraction of data into the record byte streams.

4.3.1 Conduit Entry Points

The conduit needs to interact with the HotSync Manger. The HotSync Manger calls some entry points in the conduit. There are three types of entry points that the HotSync Manager calls in your conduit:

- Required entry points that must be implemented in each conduit
- Optional entry points that provide additional information and/or capabilities
- File-linking entry points that you can implement to support file linking in your conduit

The following table provides a brief description of these conduit entry points.

Table 2 Conduit Entry Points Called by the HotSync Manager

Function Type	Function Name	Description
Required	IPDClientNotify-> BeginProcess	The main conduit entry point.
Required	IPDClientNotify-> GetConduitInfo	Returns information about the conduit to the HotSync Manager (based on parameters passed in). Return Information like the conduit's name, version number for display purposes.
Customization (optional)	IPDClientNotify-> ConfigureConduit	Presents a dialog that allows the user to configure the conduit when a user selects

	(prior to HotSync 3.0.1) or IPDClientNotify-> CfgConduit (HotSync 3.0.1 or later)	their conduit from the HotSync Manager **Custom...** menu. Palm strongly recommends that you provide this function

The BeginProcess method, the main entry point, is to notify client modules that they can run their conduit logic. The BeginProcess should return False when the conduit process is done with the server and it should continue with the HotSync Manager process.

4.3.2 The Excel File

The ScoreList handheld database is stored in memory as a file header followed by a sequence of records. The file header section of each database contains the following information about the database:

- Name
- Creator
- Type
- Number of records
- Last synchronization date
- Optional variable-length application information block
- Optional variable-length sorting information block

Each record in the database consists of the following information:

- Record ID
- Record category index
- Record attributes: indicating something like if a record is marked private, or if it is currently in use, or if it should be archived at the next synchronization or if it should be deleted.
- Record data

We do not need to update any database information from desktop. So we do not need to worry about storing the database information in desktop. But we need to store any record information in desktop, in our case, the Excel file.

We store each piece of information in the Palm records into a column in the Excel file. For example, we store and display the NAID in the NAID column, the last name in the Last Name column, the category in the Section column, etc…

However, some information, like the unique record id and the category number, are not appropriate to be shown (users may be confused if shown). So they are stored in a hidden column (First column A). There is an additional column in the file called Average, which shows the average score that is automatically calculated from the scores in each record.

4.3.3 Implementation of the IPDClientNotify Interface

To implement the conduit is actually to implement the IPDClientNotify interface. Since the conduit is based on the COM Sync suit, you need to add the following references to your visual basic project:

- Palm ComDirect 1.0 Type Library
- Palm ComStandard 1.0 Type Library

The conduit visual basic project contains 5 classes, CNotify, CDtCategories, CDtData, CScoreReord and CSync. Of all 5 classes, CNotify directly implements the IPDClientNotify interface while the other four classes are auxiliary classes.

CNotify Class

This class directly implements the IPDClientNotify interface. It implements all three functions in the interface:

- IPDClientNotify_BeginProcess,
- IPDClientNotify_CfgConduit,
- IPDClientNotify_ConfigureConduit,
- IPDClientNotify_GetConduitInfo.

CDtCategories Class

This class is an implementation of categories. It contains an array of category items and each category item is defined as a user-defined data type. They are defined as in the following:

```
'   Category item declaration
Private Type CategoryItem
        pName As String * 15    'category name
        nId As Byte             'category id
        bDirty As Boolean       'the dirty flag
```

```
End Type

Private colCats(0 To 15) As CategoryItem    'total 16 categories
```

It then contains methods for you to get/set category name, id, to convert categories from the palm format to the CategoryItem format and vice versa.

CScoreReord Class

This class implements the palm record structure on desktop. It only contains properties but no methods. The following table lists the properties and their corresponding fields in palm.

Table 3. The Properties of the CScoreRecord Class

Property of CScoreRecord	**Explanation or Corresponding field in palm**
RecordId	Record unique id in palm
CatNum	Category number (0 – 15)
Attributes	Record attributes
flags	LibPackedDBRecord.flags
naidOffset	LibPackedDBRecord.naidOffset
lastNameOffset	LibPackedDBRecord.lastNameOffset
firstNameOffset	LibPackedDBRecord.firstNameOffset
scoreOffset	LibPackedDBRecord.scoresOffset
noteOffset	LibPackedDBRecord.noteOffset
Naid	Student id

LastName	Last name
FirstName	First name
Scores	Scores
Note	Note
hiddenInfo	The hidden information (like the record unique id) stored in the Excel file column A (first column)

CDtData Class

This class first defines a collection of class type CscoreRecord and an object colData of this collection data type and an object pDtCats of type CDtCategories. Then it implements various functions to operate on these two objects and the Excel file.

- OpenDatabase: Open the excel file, import all information from the file to the colData and pDtCats.
- UpdateCategoryInfo: update category information in the collection pDtCats from the excel file.
- UpdateHiddenRecordInfo: update hidden record information in the excel file from other information shown.
- CreateDatabase: create the excel file (for the first time)
- ReadRecord: read a record form collection colData and make it ready to be sent to palm
- WriteRecord: write a record of type CScoreRecord from the information sent from palm.

- MergeRecord: merge two CScoreRecord type records into one record.
- RemoveAllDeletedRecords: remove all deleted records from collection colData.
- SaveToDesktop: save all CScoreRecord records in collection colData into the excel file

CSync Class

This class implements the actual synchronization. It has the following functions:

- Synchronize: Control synchronization selection based on HotSync action
- SyncStandard: Perform normal synchronization tasks
- Sync: Implement synchronization
- CopyToPc: Copy database to Desktop
- CopyToPalm: Copy database to Palm
- LoadCategoriesFromPalm: Load categories into collection CDtCategories
- SyncCategories: Synchronize the categories
- OpenRemoteDatabase(ByVal pName As String): Open the palm database
- OpenDesktopDatabase: Open the excel file and load records into CDtData

4.3.4 The Synchronization Process

The following figure shows the whole synchronization process.

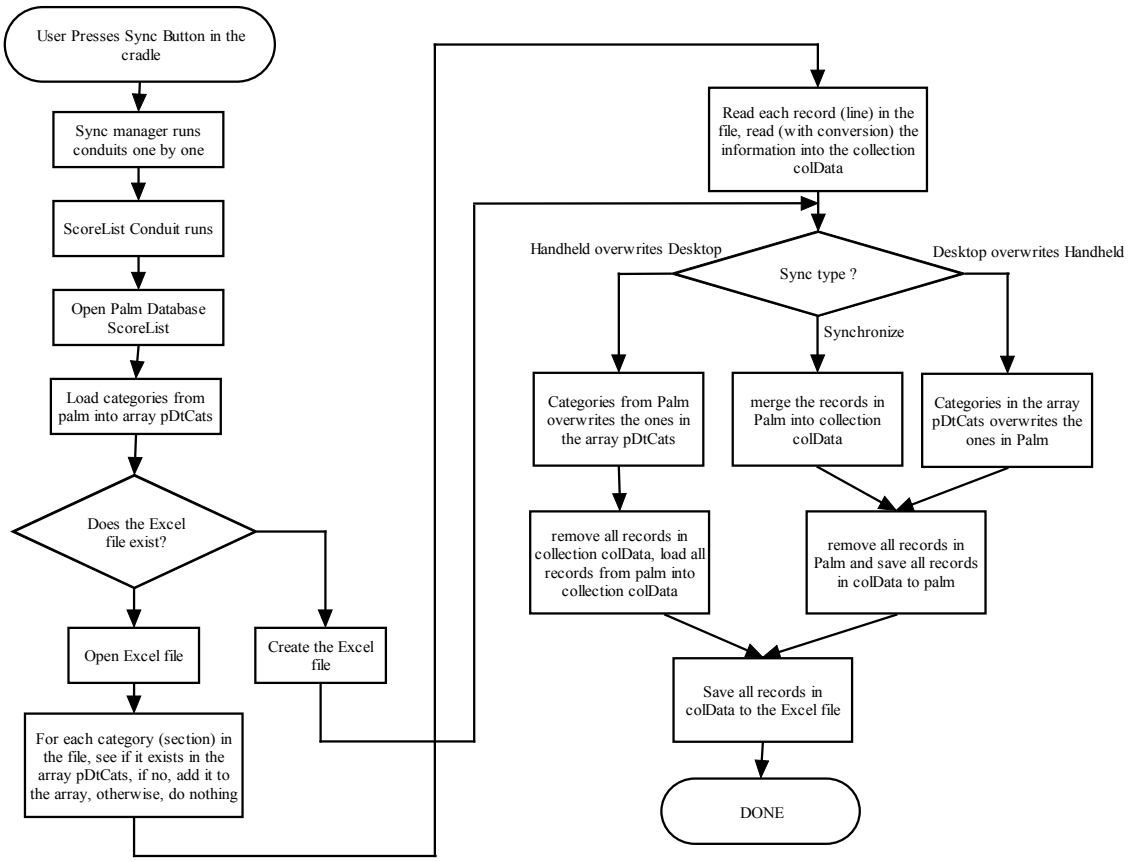


Figure 6. Synchronization Process

4.4 Software Testing

Testing is a process of executing a program with the intent of finding an error. A good test case is one that has a high probability of finding an as-yet-undiscovered error. A successful test is one that uncovers an as- yet-undiscovered error. These objectives imply a dramatic change in viewpoint. They move counter to the commonly held view that a successful test is one in which no errors are found. Our objective is to design tests that systematically uncover different classes of errors and to do so with minimum amount of time and effort.

If testing is conducted successfully (according to the objectives stated previously), it will uncover errors in the software. As a secondary benefit, testing demonstrates that software functions appear to be working according to specification, that behavioral and performance requirements appear to have bee bet. In addition, data collected as testing is conducted provide a good indication of software quality as a whole. But testing cannot show the absence of errors and defects, it can show only that software errors and defects are present. It is important to keep this (rather gloomy) statement in mind as testing is being conducted.

4.4.1 Two Testing Methods

There are two common testing methods that we usually apply.

Unit Testing

Unit testing focuses verification efforts on the smallest unit of software design - the software component or module. Using the component—level design description as a guide, important control paths are tested to uncover errors within the boundary of the module. Unit testing is normally considered as an adjunct to the coding step. After source level code has been developed, reviewed, and verified for correspondence to component level design, unit test case design begins. A review of design information provides guidance for establishing test cases that are likely to uncover errors in each of the categories discussed earlier. Each test case should be coupled with a set of expected results. Unit test is white—box oriented, the step can be conducted in parallel for multiple component.

Integrated Testing

After we apply unit test on all modules we developed in our program, we can make sure that they all work fine individually. Then we need to put them together to make sure it function correctly. The reasons we need the integration test are that data could be lost across an interface; one module can have an inadvertent, adverse affect on another; sub functions, when combined, may not produce the desired major function; individually acceptable imprecision may be magnified to unacceptable levels; global data structures can present problems.

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The object is to take unit tested components and build a program structure that has been dictated by design.

There is often a tendency to attempt non—incremental integration. All components are combined in advance. The entire program is tested as whole. And chaos usually results. A set of errors is encountered. Correction is difficult because isolation of causes is complicated by the vast expanse of the entire program. Incremental integration is the antithesis of the big bang approach. The program is constructed and tested in small increments, where errors are easier to isolate and correct; interfaces are more likely to be tested completely; and a systematic test approach may be applied.

4.4.2 Testing Procedures and Results

We have done both unit testing and integrated testing. Unit testing is done on the Palm Simulator. Integrated testing including the conduit testing is done on a real Palm m505 color PDA.

Each testing and its goals, procedures and results are listed in the following.

List/Enter/Edit Student Information and Scores Testing

This testing is to make sure that we can list student information (NAID, Last Name and First Name) and scores correctly, enter a student record, enter a score, edit a student record and delete a student record. We checked the following items:

- Pressing the New button on the Score List form brings users to the Record Edit form
- The user is able to enter NAID, last name and first name using any input methods, i.e. Graffiti or soft keyboard.
- When entering information, if the screen cannot show all information, it scrolls automatically.
- Pressing the Done button on the Record Edit form closes the Record Edit form and returns to the Score List form and a new record with the information just entered shows in the Score List form.
- Pressing the Details button on the Record Edit form will bring up a standard palm record details screen. On this screen, you can assign the record to a category, mark it private, enter a note and delete the record.
- Pressing Details button on the Record Edit form allows you to enter a note for the record
- You assign each record to a category directly on the Record Edit form by choosing one in the category pull down list.

- Pressing the Show button on the Score List form comes up a form with 4 options which allows you to show the records in 4 different ways, i.e. Name/NAID, NAID/Name, NAID only, Name only. If it is shown as Name/NAID and Name only, records are sorted alphabetically by the name; if it is shown as NAID/Name and Naid only, records are sorted alphabetically by the NAID
- You list records that belong to a specific category by choosing a category in the Category pull down list on the Score List form. The All category allows you to list all records. You can also create/edit category by choosing Edit Categories… in the list.
- Tapping on the title section on each record brings up the Record Edit form so that you can edit that record you selected.
- Tapping on the last column (the Note column) brings up the standard Palm Note form that allows you to enter a note. The Note form's title should be the student name.
- Tapping on the Scroll Down button lets you see more records available. Tapping on the Scroll Up button lets you see previous records.
- Tapping on one of the three score columns brings up a Pull Down list with 3 score levels (0,1,2). After you choose one score level, that score level is shown on the score column correctly.
- Three score columns can be shown on the screen one time.
- Pressing the Scroll Right button allows you to enter more scores for the student. If previously, the first, second, third score columns were shown, after you press the

Scroll Right button, you will see the fourth, fifth and sixth score columns. The column title indicates which score you are viewing.

- Pressing the Scroll Left button allows you to see the previous scores for the student. The system will move back three scores each tap.

Menu/Auxiliary Functionality Testing

The ScoreList program also has menus that provide you some auxiliary functionalities. To test these functionalities, we test the following items:

- On the Score List form, tap on the Properties silk button you will see two menus which have the following menu items: Beam Category, Font, About.
- The Beam Category menu item is not implemented, nothing will happen when you choose it.
- The Font menu item allows you to change the font type and size of the information shown on the screen.
- The About menu item allows you to show the program information (i.e. the author, version, etc..)
- On the Record Edit Button, tap on the Properties silk button you will see three menus which have the following menu items: Beam Record, Delete Record, Duplicate Record, Attach Note, Delete Note, Undo, Cut, Copy, Paste, Select All, Keyboard, Graffiti Help, Font and About ScoreList
- Beam Record menu item is not implemented. Nothing will happen when you choose it.
- Delete Record menu item allows you to delete the record you selected

- Duplicate Record menu item allows you to duplicate the record you selected, the duplicated record has "(Duplicate)" appended in the last name column.
- Attach Note menu item allows you to enter a note for the record
- Delete Note menu item allows you to delete the note for the record
- Undo menu item allows you undo what you just did
- Cut menu item allows you to cut a selected string
- Copy menu item allows you to copy the selected string
- Paste menu item allows you to paste the copied string
- Select All menu item allows you to select the whole field
- Keyboard menu item brings up the soft keyboard
- Graffiti Help menu item brings up the Graffiti symbols
- The Font menu item allows you to change the font type and size of the information shown on the screen.
- The About Score List menu item allows you to show the program information (i.e. the author, version, etc.)

Integrated Testing/Conduit Testing

When the software was finished, we did an integrated testing on a real Palm M505 PDA. We tested the whole input-output system, from user inputting information to seeing information on an Excel spreadsheet. We repeated pretty much the same testing for the Palm program as the unit testing. However, we concentrated on the conduit testing since it is a critical process to make sure the information collected by Palm is successfully transmitted to desktop.

1. Transmitting Records from Palm to Desktop

First, we entered a bunch of records including NAID, last name, first name, scores (all levels and no score) and notes. On the Desktop, we set HotSync action to Synchronize the files. Then, after the first synchronization was done. We checked the following items:

- A file called ScoreList.xls was created in the palm user directory
- Opened this file, checked all records are transmitted without losing any data
- Checked records are consistent, i.e. one record information is not misplaced to another record
- Checked each record category information is consistent with that in Palm.

2. Synchronization with HotSync Action Desktop Overwrite Palm

In this testing, we tested if records can be transmitted from Desktop to Palm. We left some record on Palm. Then in the Excel file, we entered a bunch of different records. Then after the synchronization was done, we checked the following items:

- All records originally in Palm were gone
- All records on the Excel file were in the Palm
- Checked each record was consistent between Palm and Desktop

3. Synchronization with HotSync Action Palm Overwrite Desktop

In this testing, we tested if records can be transmitted from Palm to Desktop and be able to overwrite the ones on the desktop. We left some records in the Excel file. Then

we entered a bunch of different records on Palm. Then after the synchronization was done, we checked the following items:

- All records originally in the Excel file were gone
- All records Palm were in the Excel file
- Checked each record was consistent between Palm and Desktop

4. Synchronization with HotSync Action Synchronize the Files

Type of Synchronization can keep all records in both Palm and Desktop after synchronization. Usually, for the same record (determined by the record id), information on Palm will overwrite that on Desktop. However, if the number of scores for the record on Desktop is larger than that for the same record on Palm, the record on Palm will be replaced by the one on Desktop.

First, we entered records in both Palm and Desktop with some records the same (with same record id), some records different, some records on Desktop have more scores on the same ones on Palm. Then, after the synchronization is done, we checked the following items:

- All records on Palm and Desktop were kept
- The records with more scores on Desktop replaced the corresponding ones on Palm, otherwise, the records on Palm overwrote the corresponding ones on Desktop

REFERENCES

[1] Foster, Lonnon R, Palm OS programming Bible, IDG Books Worldwide, Inc., 2000.

[2] Pressman, Roger S., Software Engineering, fifth edition, McGraw-Hill Higher Education, 2001.

[3] CodeWarrior Manuals for CodeWarrior for Palm OS Platform, Metrowerks, A Motorola, Inc. Company, 2000.

[4] Palm OS Programmer's Companion, Palm Inc., 2000.

[5] Palm OS SDK Reference, Palm Inc., 2000.

[6] CodeWarrior Targeting the Palm OS Platform, Metrowerks, A Motorola, Inc. Company, 2000.

[7] Palm OS Programming Development Tools Guide, Palm Inc., 2000.

[8] Code Warrior Constructor for Palm OS Platform, Metrowerks, A Motorola, Inc. Company, 2000.

[9] Introduction to Conduit Development, Conduit Development Kits for Windows and Macintosh, Version 4.02, Palm Inc., 2001.

[10] COM Sync Suit Companion, Conduit Development Kits for Windows, Version 4.02, Palm Inc., 2000.

[11] MSDN, Microsoft Developers Network, http://msdn.microsoft.com/, Microsoft Corporation, 2002.

APPENDIX A: Software Manual

A.1 Installing The Software

1. Hardware Requirements:

- A Palm Handheld device running Palm OS 2.0 or later
- A Desktop computer running Microsoft Windows 95 or later with USB port
- A Palm Cradle and USB cable

2. Software Requirements:

- Palm Desktop software (Including HotSync Manager and Installer)
- Microsoft Excel application

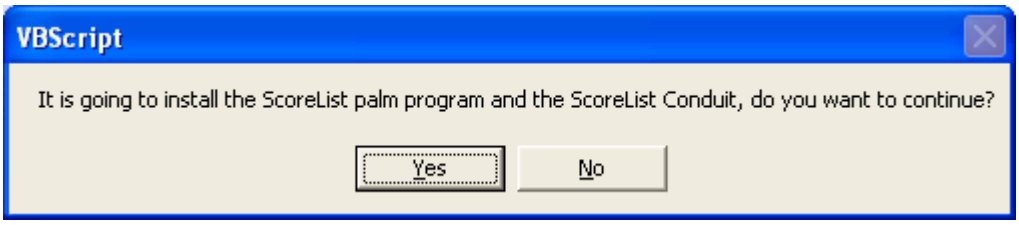
3. Install

Insert the install CD into your CD ROM drive. Suppose your CD Rom drive is D and you have successfully installed the Palm Desktop software or the HotSync Manager on your machine. Following the following steps to install the software.

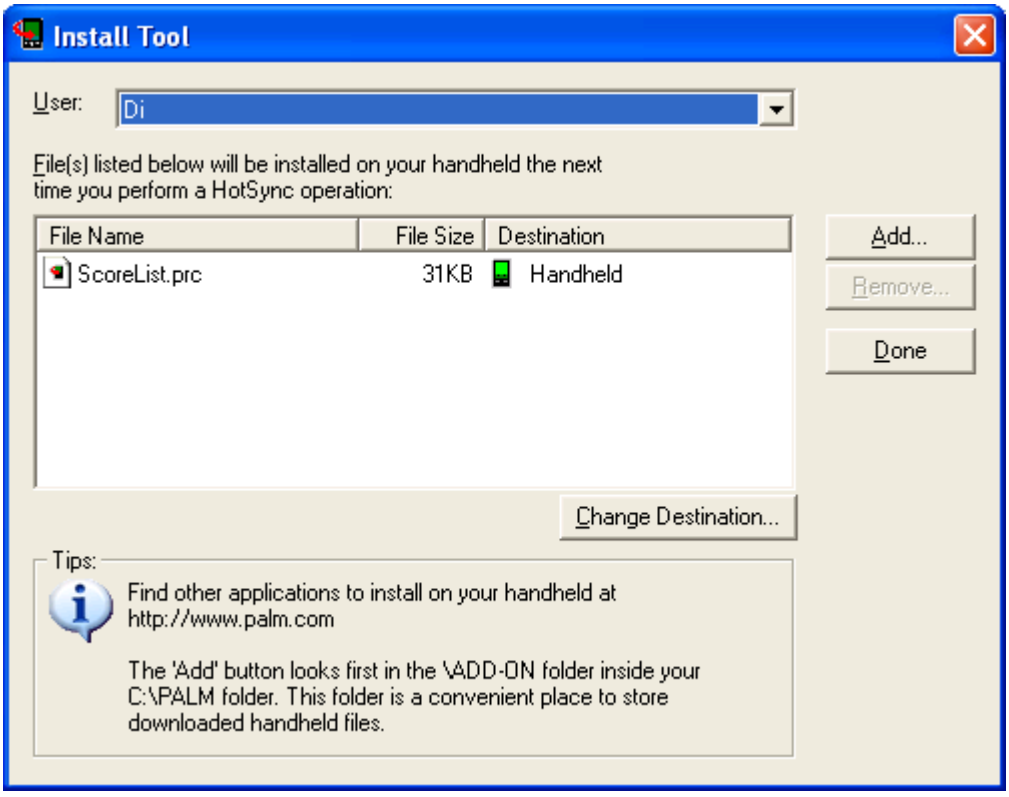
- Create a directory with any name. Suppose you create a directory called C:\ScoreList.

- Copy the files in D:\Installer to C:\ScoreList.

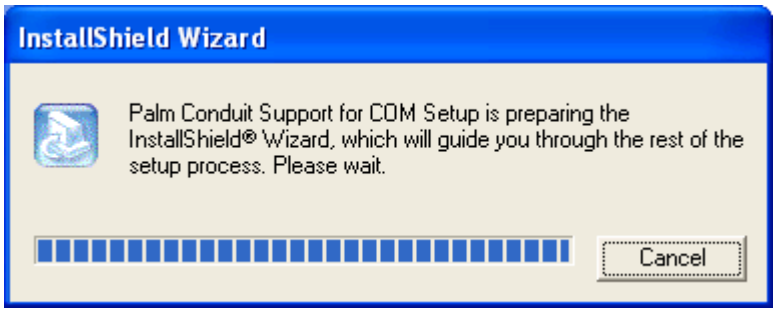
- Run the install program install.vbs in C:\ScoreList. First you will see the following screen.



Click Yes to install the software, click No to quit. If you click Yes, then you will see the following screen:

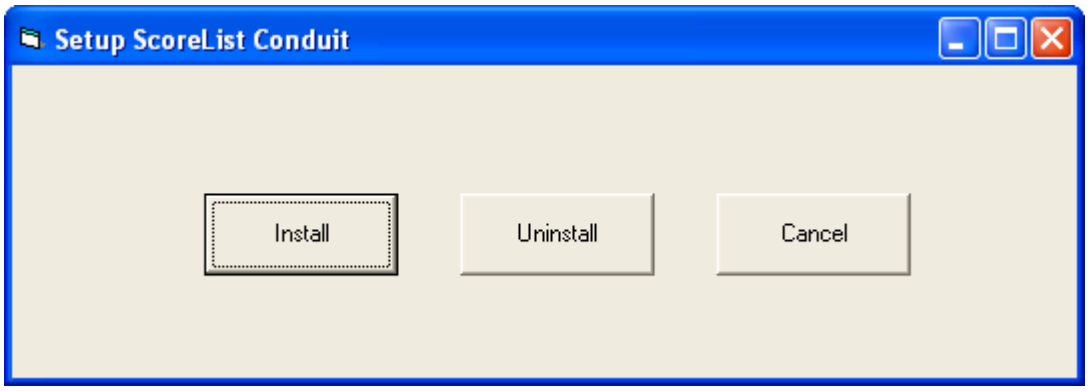


Click Done to install ScoreList.prc on your handheld the next time you perform a HotSync operation. Then you should see the following the following screen quickly passed by (if your machine is really fast, you may not see this screen)



Next, you will see the following screen after a while.

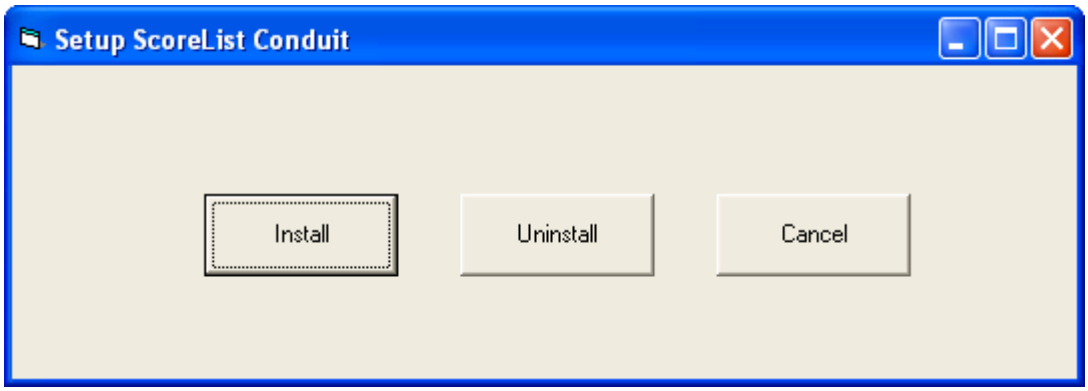
It tells you that you have successfully registered the ScoreList conduit DLL. Click OK to continue. Last, you will see the following screen:



Click Install to configure the conduit on your machine. Click Cancel to quit the program without configuring the conduit this time. (You have to run the install program again to configure the conduit). Click Uninstall to remove the setup of the conduit on your machine.

4. Uninstall

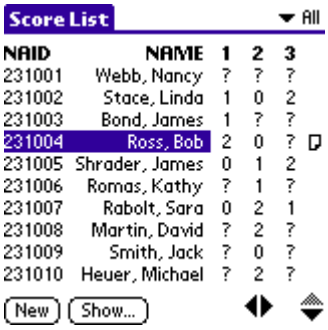
Suppose you have installed the software on C:\ScoreList. Run the uninstall.vbs on that directory. When you see the following screen



Just click the Uninstall to uninstall it.

A.2 How to use the software

The palm program works like a spreadsheet. It has rows and columns. There are column NAID (student id), name (last name and first name), and up to 255 numbers of score columns and a note column to record anything about the student. Records can be grouped into different categories, which are customizable. For example, a class may have 3 sections, Section A, Section B and Section C. You can assign each student into its appropriate sections.



How to create a new record and edit a record?

On the Score List form, to create a new record, press New button. To edit a record, tap on the Name or NAID column of the record. In Both cases, you will see the Record Edit form as in the following:

Then, you can add/edit Naid, Last Name and First. When you are done, Press the Done button to exit.

How to browse records and scores?

There are four scroll buttons, Scroll Left, Scroll Right, Scroll Up and Scroll Down, for you to browse records and scores.

- Scroll Left: clicking this lets you see earlier scores, ex. If you are on the 4th score, you can see 4th - 6th scores. Clicking this allows you to see the 1st – 3rd scores
- Scroll Right: clicking this lets you see later scores
- Scroll Up: clicking this lets you see previous records
- Scroll down: clicking this lets you see next records

How to enter a score?

Scores are 3 levels, 0 1 or 2. "?" means no score. To enter a score, simply tap on one of the score columns, you will the following list.

Then, just pick one from the list.

How to show records differently?

On the Score List form, press the Show… button, you will see the following form:

Pick one and press OK and you are done. If you pick “Naid,Name” or “Naid Only”, records will be sored by Naid; if you pick “Name,Naid” or “Name Only”, records will besorted by Name.

How to use menus?

On the Record Edit form, type the Properties silk button, you will be able to see the following menu:

You can delete a record, duplicate a record, etc.. Please note that Beam Record is not implemented.

How to enter a note?

There are several ways to enter a note for a record. First, you can tap on the Note column on the Score List form. Second, you can press Note button on the Record Edit form. Third, you can use menu to attach a note. The note form is shown as in the following figure:

The title of the note shows the student name.

How to delete a record?

There are two ways to delete a record. One is that you press Details button on the Record Edit form, you will see the following form:

Then on this form, click Delete… button to delete the record. The other way is through menu, selecting Delete Record….

How to mark a record private so that users without a password will not see it?

On the Record Details form, check Private checkbox, the record will be marked private so that users without security will not see that record.

How to put a record into a different category?

On the Record Edit form, select a category from the upper left corner category list and the record will be assigned to that category.

Or on the Record Details button, you can select a category for the record.

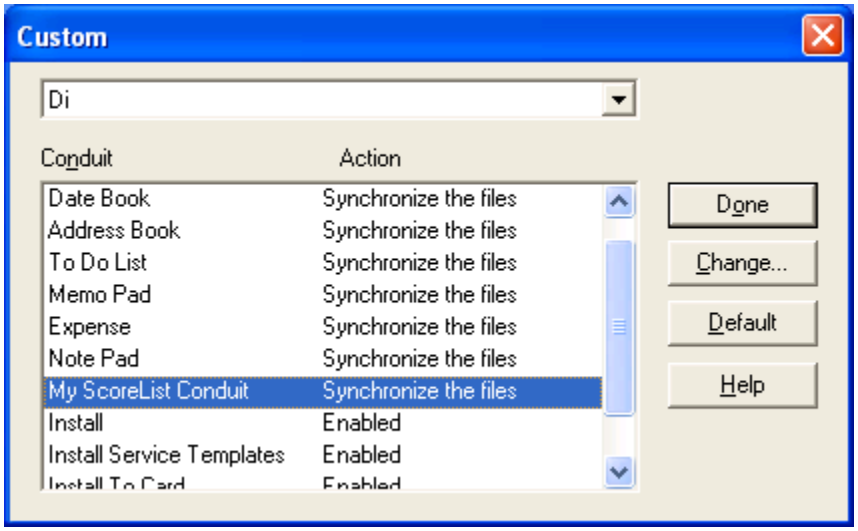
How to list records in a specific category?

On the Score List form, simply choose from the category list on the left upper corner.

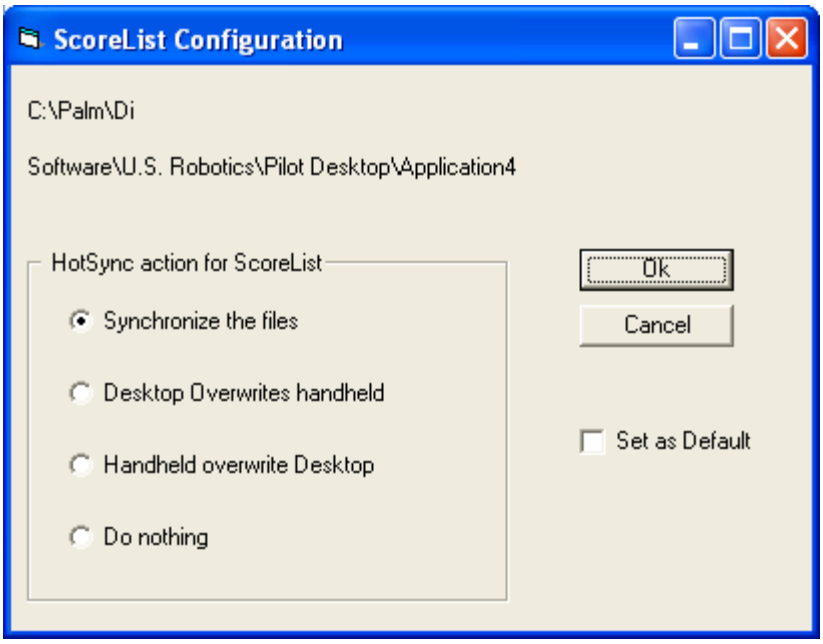
After selecting a category, only records with that category will be shown.

How to synchronize data with desktop

Once you have installed the conduit successfully, you will be able to synchronize it just like other programs. On the Custom window in the HotSync Manager:



You configure your ScoreList conduit by clicking the Change… button. The following screen appears.



On this screen, you can choose the HotSync action type and set it as a default action by checking the Set as Default checkbox. Then all the future synchronization will use that type.

After the configuration is done, all you need to do is pressing the Sync button in the cradle. The first time you synchronize, it will generate a file called scorelist.xls on your palm user directory (If you install Palm Desktop by default, your palm user directory will be "c:\palm\<your windows account name>"). It is a standard Microsoft Excel file. It contains all records from palm.

What HotSync Action should I choose?

Here are a few instructions:

- To keep all changes you made in either Palm or Desktop available to both places, choose Synchronize the files. But you need to know when the same record is changed in both places, the palm record overwrites the desktop one with one exception: when the number of scores in the Desktop record is larger than that in the Palm record, the desktop record overwrites the one on Palm.
- If you only want to keep the records on Palm, choose Handheld overwrites Desktop.
- If you only want to keep the records on Desktop, choose Desktop overwrites Palm
- If you don't want to synchronize ScoreList database in the next synchronization, choose Do Nothing.

APPENDIX B: Source Code

The source code is not printed. It is stored in the accompany Compact Disk. Suppose D is the CD-ROM drive, D:\ScoreList stores all the source code for the ScoreList palm program, written in CodeWarrior; D:\ScoreList_Conduit stores all the source code for the ScoreList Conduit program, written in Visual Basic; D\ Install_Conduit stores the source code for the program to configure the ScoreList Conduit as part of the install program; D:\Install_PalmCom stores all the components needed to run the conduit from Palm Inc.