RCU

Version 2.5 User's Guide

NMI & Revelar, Inc. License Agreement

Before using this software, please read the License Agreement (“Agreement”) text file in the RCU folder you now have installed on your computer. Your use of the software implies that you agree to be bound by the Agreement. If you do not accept the Agreement, promptly return the product for a full refund (less shipping).

RCU was written by Van C. Evans and W. Bryant Eastham. Documentation was written by Sally Miller and Van C. Evans.

Translation was provided by Bernd Schenk, Ralf Hinz, Lars Oro, Bernard Le Due, Alejandra Lugo, and Robyn Buckwalter.

Thanks also to Andrea Evans, Annette Eastham, Robyn Buckwalter, Joe Patterson, Steve Clark, Ed Firmage, and Carol Schuer of Apple Computer, Inc.

Technical Support

Please read carefully the instructions in this guide. It was written to be straightforward and intuitive. If you experience any problems in the installation process or the normal use of the software that we have not explained, feel free to call. We are committed to supply telephone support free of charge, *provided* you return your completed registration card. You may call Monday through Friday from 9:00 a.m. to 5:00 p.m. Mountain Time, at these numbers:

800-669-5191
801-278-7102

Have your serial number handy, and where possible, be seated at your computer with the Newton PDA ready. We have found that we can help you best when we walk through the problem right on your screen. It is not uncommon for us to take your name and number and then return your call, when someone is available to help you. If this is the case, please be patient with us, since we will return your call as soon as possible. You may also email at:

support@revelar.com

Contents

WELCOME **6**
- Features For Version 2.5 6
- About This Guide 7

BASIC USER INFO **8**
- Installation 8
- System Requirements 8
- Registration 9
- Connecting to the Macintosh 10
- Disconnecting 12
- Connection With Batteries 12
- Main Window 13
- Title Bar 13
- Icons 14
- Status Text 15
- Pop-Up Menus 15
- Field Editor 16
- Understanding the Overview 17
- Changing View by Folder 17
- Selecting a Different Overview Type 17
- Performing a Find 18
- Changing the Range 19
- Working with Entries 19
- Adding Entries 20
- Adding Fields 21
- Navigating through Fields 21
- Conditional Fields 22
- Specifying Class Type 22
- Editing Fields 22

Locked Applications and Fields 22

Saving Changes to Your Newton 23
Printing 23
Changing Applications 23
Setting Preferences 24
Connection 24
Palette Preferences 24
Printing 25
Importing and Exporting 25
Exporting 26
Importing 27
Exporting and Importing Note Fields 28
Saving Templates 30
REFERENCE INFO 31
File Menu 31
Connect/Disconnect 31
Abort 31
Get Info… 31
Get Screen Shot… 32
Set Time & Date 32
Download Package… 32
Import/Export XXXX… 33
Close 33
Page Setup… 33
Print… 33
Quit 33
Edit Menu 34
Cut, Copy, and Paste 34
Preferences… 34
Localization… 35
Application Menu 35
Action Menu 36
Add XXXX... 36
Delete XXXX... 36

Save 36
Save As… 36
Add Field... 37
Delete Field 37
Find 37
Find All 37
Select Range... 37
Select Overview.. 37

Calendar 38

ADVANCED USER INFO 39
Overview of the ADF File Format 39
General Format 40
Application Syntax 40
Soup Syntax 41
Overview Syntax 42
Entry Syntax 42
Class Syntax 46
Range Syntax 46
Extension Syntax 47
Example ADF File 47

Customizing RCU 59
Creating Custom Overviews 59
Changing the Field Order 60

Welcome

Thank you for choosing the Revelar Connection Utility™ (RCU) for the Macintosh® and Newton systems. RCU gives you a real time connection to the applications of your Newton PDA from your Macintosh or Windows computer.

Features For Version 2.5

- Add, delete, browse, modify, and print your Notes, Names, Calendar, To Do items, Calls (write protected).
- View custom overviews, view by folder, and use the Find command to search for any text in the selected application.
- Download packages by connecting to the Connection application or to RCU.
- Get information about your Newton, such as owner information, system information, memory usage and availability, soups (data files).
- Picture support. You can sketch a picture on your Newton and RCU will bring it up to the desktop. From there you can copy and paste it to the scrapbook, a paint/DTP program, etc. You can also do just the opposite. Take any graphic from a paint program (even a color PICT or a photo), paste it into the RCU's picture field and RCU will convert it to a Newton picture. Pictures are supported in *any* note field, which includes notes in Notes, Names, and Calendar.
- Synchronize the Date and Time of your Newton with your Macintosh.
- Newton screen shot. Take a snapshot of the current Newton screen and then copy and paste it to any other Macintosh program. If the RCU connect window on the Newton is in the way, just close it. Closing the window does not interrupt connection.
- Import/Export. This version supports Notes, Names, and Calendar. To Do and Calls are export only.
- Third-party software support. Developers: write an "ADF" file in just a few hours and let RCU be your connection to the desktop.
- Localized in English (USA), Spanish, German, and French.

About This Guide

This guide is divided into three parts: Basic User Information, Reference Information, and Advanced User Information. The first part of this guide explains how to install and register RCU and connect your Newton to the desktop. It acquaints you with the main window of RCU (e.g., the icons, status bar, and pop-up menus, etc.) and explains how the overview function is used to display information (entries). You learn how to add and/or change fields to an entry, import and export files, set preferences, download a package, and perform other functions.

The second part of this guide provides descriptions of the menu items and windows accessible from pull-down menus. This section explains many features that aren't discussed in section 1, such as taking a screen shot and downloading a package. The third part of this guide explains how to extend RCU by creating ADFs for other Newton software and customizing overviews.

Basic User Information

Installation

You can install the RCU2.3.pkg on your Newton PDA by using one of the following:

- Newton Backup Utility (NBU)
- Newton Connection Utility (NCU)
- Newton Package Installer (NPI)
- RCU 2.x or greater

If you choose RCU to install the first time, you must connect to the "Connection" application on the Newton. After that, you can connect to RCU on the Newton to install other packages. You must have a standard Macintosh serial cable. The cable that came with your MessagePad will do.

If you intend to have RCU connected for an extended period of time, we recommend you plug your Newton device into an AC power adapter. If you do not, keep in mind that your Newton PDA will shut off at the sleep interval you set in preferences if there is no activity in RCU for that interval. Simply being connected does not keep your Newton awake.

System Requirements

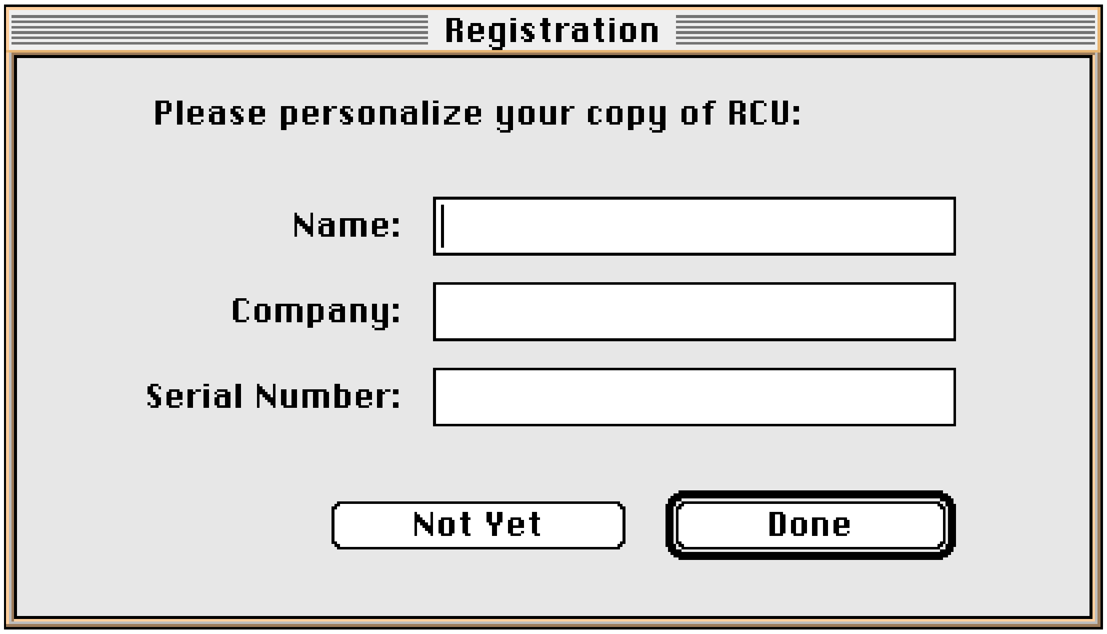
- Macintosh with 68040 processor or greater. Power PC compatible
- 8 Mb RAM
- System 7.1 or greater
- Apple Modem Tool (place this in your extensions folder)

Registration

To register your copy of RCU:

1. Launch RCU 2.5 on your Macintosh.

The Registration window opens as shown on the next page:



2. Enter your name, company (if applicable), and serial number.

The serial number is a license to use RCU on a single Newton PDA. If you want RCU installed on another Newton PDA, you must buy another copy. Enter your serial number in upper case letters. Your serial number is found on the bottom of your RCU box as well as the installation card.

3. Press the Done button to close the Registration window.

If you press the "Not Yet" button, you will still be able to connect and download packages, but will not be able to retrieve any data from your Newton PDA.

Connecting to the Macintosh

This section assumes you have read and understand the Newton's basic operations and have already downloaded the RCU2.3.pkg file into your Newton device and registered the software. If you have not already done this, refer to "Installation" and "Registration" in this guide.

1. Plug one end of your serial cable into the modem or printer port of your Macintosh and the other end into your Newton.

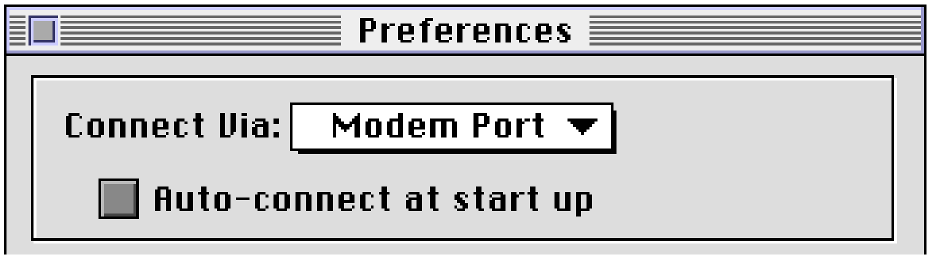
The cable end is cylindrical with one flat side, or has arrows pointing up. Place the flat side up to plug in the cable. You should have already accomplished this step to download the RCU2.3.pkg. We recommend you use the modem port, since you may wish to print reports while using RCU.

You must have the Apple Modem Tool installed in your System Folder (Extensions folder) and you should turn AppleTalk off the port you are trying to connect to. The Apple Modem Tool is found on the NBU or NCU disk that came with your Newton PDA. You turn AppleTalk off through the Chooser.

2. Double-click the RCU icon from the Finder to launch the program on your Macintosh, if it not already launched.

You must have at least 4 Mb RAM available on your Macintosh to properly use RCU. RAM requirements may be more if you have a high-resolution/color monitor.

3. Select Preferences… under the Edit menu to open the Preferences window.



Connect Via: Modem Port

4. Select the desired serial port on the Connect via: pop-up menu.

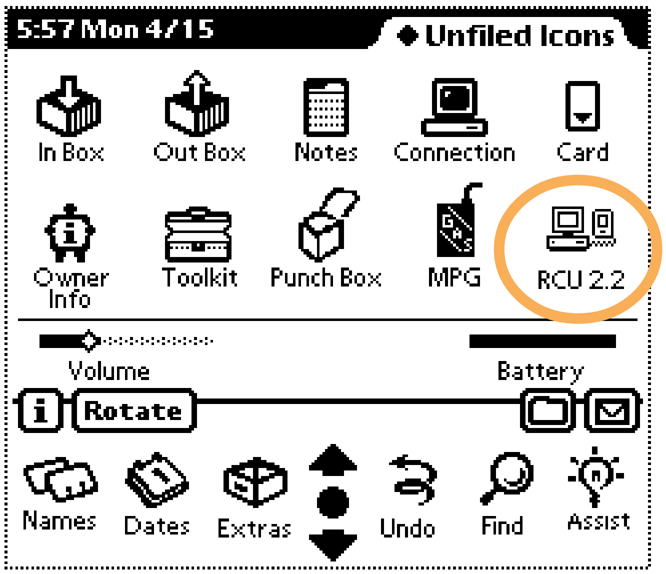
Make certain the port you select is the same port you have plugged your serial cable into. The next time you launch your Macintosh RCU program, the Connect via: menu item will default to the last selection you made.

From the Preferences window, you may also check on the Auto-connect at start up button. With this preference, RCU automatically attempts connection to your Newton as soon as you launch the application. You will still need to press the connect button on the Newton side.

5. Close the Preferences window.

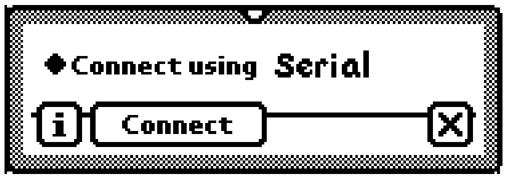
6. Turn on your Newton, if it is not already on, and tap the Extras button to open the Extras drawer, if it is not already open.

7. Tap the RCU 2.3 icon to launch the program on your Newton device.

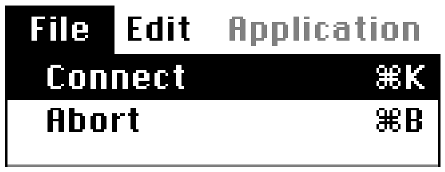


Trouble Shooting: If you do not see the RCU 2.3 icon, try reinstalling the application. If you installed RCU 2.3 on a PC card, make sure that the card has not been removed.

Once you launch RCU 2.3 on your Newton, the following window displays:



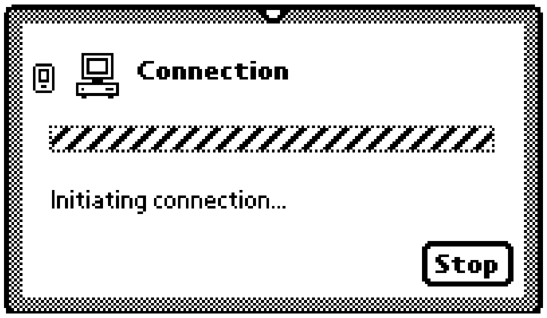
8. Select Connect from the File menu on your Macintosh.



The status text on your Macintosh changes to: Waiting for connection….

9. Tap Connect on your Newton to start the connection.

A window on your Newton shows the connection progress.



Initiating connection...

Once the connection is made, the status text on your Macintosh changes to: You are connected to RCU. The status text and a window on your Newton informs you if a connection isn't made.

Note: You will note that the Newton performs sluggishly while connection is made.

Trouble Shooting: If a connection is not made, check that the cables are properly attached and make sure no other programs are tying up the port. AppleTalk must be inactive for the port you are trying to connect to (AppleTalk is inactivated through Chooser), and the Apple Modem Tool should be installed in your System Folder (Extensions folder). If you have a Duo or PowerBook with an internal modem, you may have to turn off the modem first.

Disconnecting

To properly disconnect the link between your Macintosh and Newton, either select Disconnect from the File menu or press the Connect/Disconnect icon, or press the Disconnect button on your Newton. You will note that the Newton performs sluggishly while connection is made.

You need not select Disconnect on both your Macintosh and Newton. Disconnecting from one or the other sends a signal for both to disconnect. Unlinking may take a few seconds.

Connection With Batteries

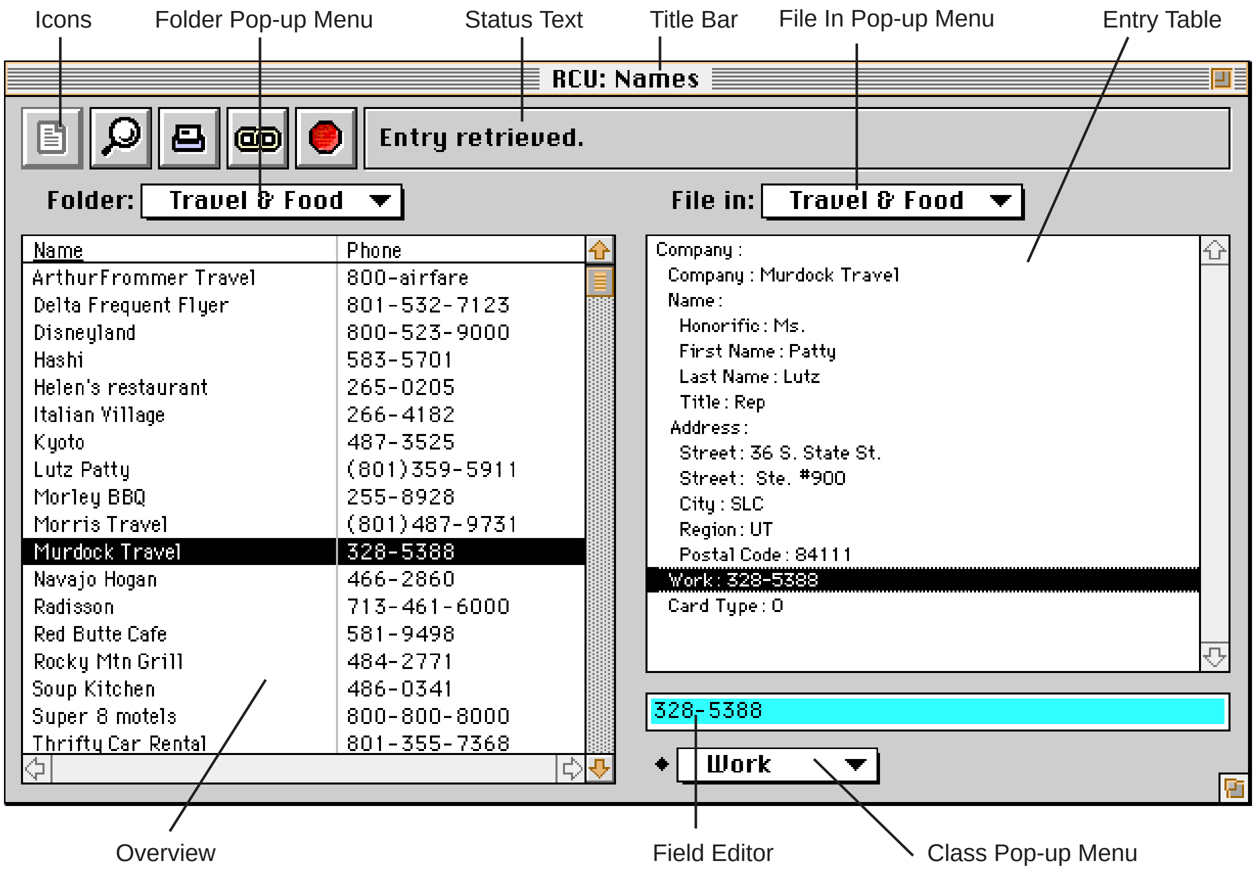
If your Newton is running on batteries and is set to "sleep" within a designated time (i.e., after 1, 5, or 10 minutes of inactivity), a connection to RCU won't necessarily keep your Newton on if you are not actively using RCU. Unlike your Macintosh, which stays on until you turn it off, the sleep function turns off your Newton when it is not active in order to conserve battery power. While you use RCU on your Macintosh, such as adding, import, or modifying entries to your Newton, the sleep function on your Newton will not turn on. The sleep function also won't go into effect if your Newton is plugged into an A/C adapter.

Main Window

This section explains the components of the RCU main window, including the title bar, icons, status bar, field editor, and pop-up menus.

Note: This guide assumes you are familiar with your Newton system. An explanation of the built-in applications is not given here, only a description of how RCU interacts with them.

Once you have linked up, RCU gives you access to the Newton's built in applications of Names, Notes, Calendar, etc. The main window of RCU, which appears on your Macintosh, displays the information that is stored on your Newton . This is a "virtual" window, which means that no information is stored on your Macintosh.



Entry retrieved.

Name	Phone
ArthurFrommer Travel	800-airfare
Delta Frequent Flyer	801-532-7123
Disneyland	800-523-9000
Hashi	583-5701
Helen's restaurant	265-0205
Italian Village	266-4182
Kyoto	487-3525
Lutz Patty	(801)359-5911
Morley BBQ	255-8928
Morris Travel	(801)487-9731
Murdock Travel	328-5388
Navajo Hogan	466-2860
Radisson	713-461-6000
Red Butte Cafe	581-9498
Rocky Mtn Grill	484-2771
Soup Kitchen	486-0341
Super 8 motels	800-800-8000
Thrifty Car Rental	801-355-7368

```
Company :
  Company : Murdock Travel
  Name :
    Honorific : Ms.
    First Name : Patty
    Last Name : Lutz
    Title : Rep
  Address :
    Street : 36 S. State St.
    Street :  Ste. #900
    City : SLC
    Region : UT
    Postal Code : 84111
  Work : 328-5388
  Card Type : 0
```

328-5388

Title Bar

The title bar indicates the current application being accessed from your Newton. RCU either opens to the last application used or, upon first launching RCU, to the first application (the Calendar). If the application is locked (i.e., is read-only), then the title bar shows the name of the application in brackets. You use the Applications pull-down menu to select (open) a different application from your Newton.

Icons

Text/Picture

This icon is disabled until you tab into or click on a note field in the entry overview. The note field is available in the Notes, Names, and Calendar applications. Once enabled, the icon toggles between a text and picture state. In picture mode, you can add (paste in), get, browse, or delete pictures in the Newton. When in picture mode, use the right and left arrow keys to navigate through the pictures in a note field.

Find

This icon opens the Find window, where you can search for entries that match your criterion. The Find feature only searches entries in the currently selected application and in the current overview.

Print

This icon prints the current overview, the current entry, or the field editor.

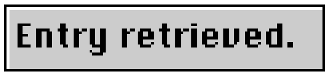
Connect/Disconnect

This icon initiates the connection to the Newton. Or, if the Newton is connected, it disconnects the link.

Abort

This icon aborts the current command or commands. You press this icon if you want to change your mind about retrieving time-intensive information from your Newton. Because commands may stack up, you may need to press this icon more than once.

Status Text

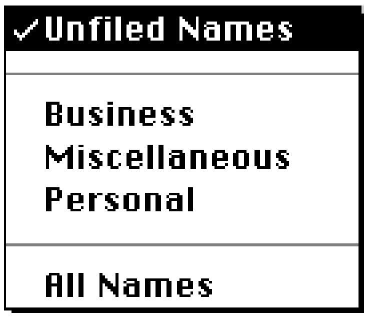


The text to the right of the icon bar gives you feedback as to what the Newton and RCU are doing. It indicates if you are connected, have received an entry, etc.

Pop-Up Menus

Folder

This menu displays the folders in your Newton that apply to a particular application. Selecting a menu item retrieves everything in that folder and loads it into the overview.

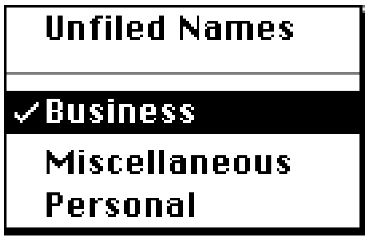


RCU always defaults to the Unfiled folder (e.g., Unfiled Names). Selecting the All folder (e.g., All Names) displays all entries in the overview. Ranges and Find work only within the entries in the selected folder (i.e., within the current overview).

The Folders menu disables for applications that don't store information in folders, such as the Calendar application, which doesn't store appointments in folders.

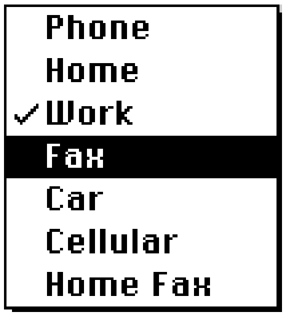
File In

This menu displays the folder where the current entry is filed. You can file the current entry into a different folder by selecting the new folder menu item and then saving the entry.



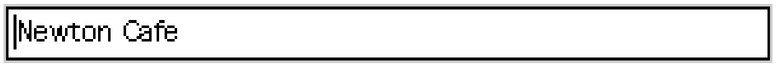
Class

This menu displays the class for the particular field you are working in. It is available for only certain fields, such as numbers. For example, you specify the class of a telephone number by clicking on the appropriate pop-up menu item.



Field Editor

This box contains the information for the current field you are working in, or is blank if you have added a new field. You add or edit the field information from this box.

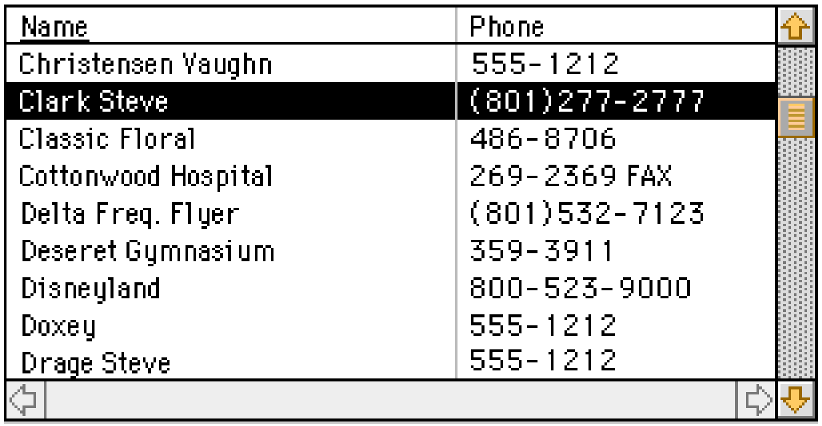

Newton Cafe

You must press enter to save additions or changes made to the field. Pressing tab deletes the field (if it is blank) and moves you to the next field. Pressing shift+tab displays the information for the previous field.

***Note*: You MUST press the ENTER key to save changes to the field, NOT the RETURN key. The ENTER key is found on the numeric keypad. Also, pressing ENTER only saves the data into the field, it does NOT save the entry out to the Newton. You must select Save from the Action menu to save the entry to the Newton.**

Understanding the Overview

Once you make a connection or display a new application, RCU automatically retrieves the overview—the list of your entries—and displays it on the left side of the screen under the Folders menu.



Name	Phone
Christensen Vaughn	555-1212
Clark Steve	(801)277-2777
Classic Floral	486-8706
Cottonwood Hospital	269-2369 FAX
Delta Freq. Flyer	(801)532-7123
Deseret Gymnasium	359-3911
Disneyland	800-523-9000
Doxey	555-1212
Drage Steve	555-1212

Since your Newton application may contain hundreds or even thousands of entries, RCU loads them in the overview only as you need them. As you scroll down the overview, RCU fills in the entries from your Newton in blocks of 25. It takes a few seconds for the Newton to send up the entries and fill them in the overview, so you may see blank areas if you scroll fast enough.

The overview reloads each time you: change the folder, change the overview type, perform a find, change the range, save an entry, or change the application.

Changing View by Folder

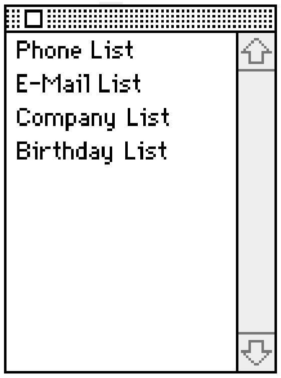
The Folders menu always defaults to Unfiled. For example, if the application is Names, the overview contains the list of Unfiled Names. If the application is Notes, the overview contains the list of Unfiled Notes, and so forth.

You use the Folders menu to change which files, or entries, are displayed in the overview. You can view All entries, Unfiled entries, or entries that have been filed to an application-specific type (e.g., the Names application allows you to file entries, or names, under Personal, Business, Miscellaneous, etc.).

Selecting a Different Overview Type

The number of columns and their titles change from application to application, and you may have different overviews within a single application. For example, you can show Names and Email addresses instead of Names and Phone numbers, as in the example above.

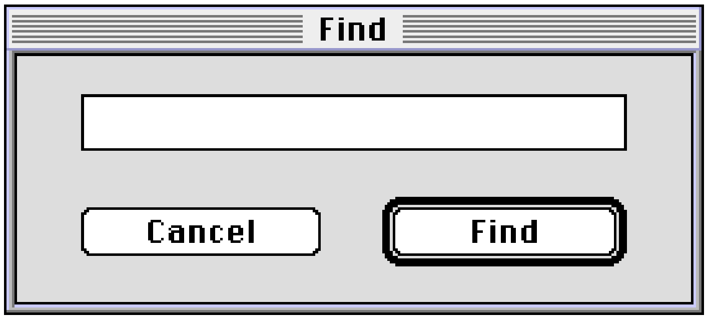
You change the overview type by either clicking on the Overview icon or selecting Select overview… under the Action menu. The list of available overview types for the current application appears in a palette:



Click on the desired overview type. The palette closes automatically. The overview reloads sorting the files under the new column type.

Performing a Find

The Find feature allows you to search for specific entries within the currently selected application. You open the Find window by clicking on the Find icon or selecting Find… under the Action menu.



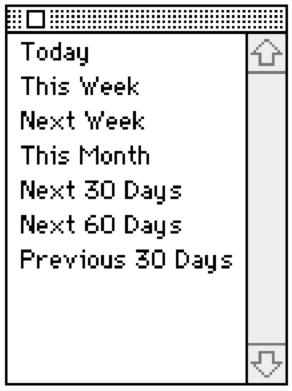
Enter in the name, date, number, or other search criteria and click on Find or press return to initiate the search. "Find" searches all entries and fields in the currently selected application and in the current overview. All entries that match your search criteria—the search results—display in the overview. The status text alerts you if no search results are found.

The Find icon 🔍 appears under the status text bar while you are in Find mode.

To clear this mode and return to the regular overview, select Find All under the Action menu.

Changing the Range

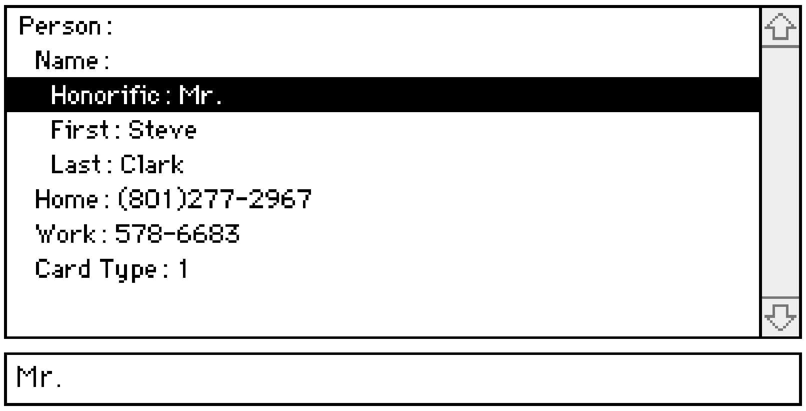
The Range palette allows you to specify a range to be displayed in your overview. For example, you may wish to see only the appointments for the current week.



You open this palette by clicking on the Range icon or selecting Select Range… from the Action menu. When you specify a range, the overview displays those entries that fall within the range. If an application doesn't support ranges, the range palette appears with no entries.

Working with Entries

Each application on your Newton contains information that is sorted into entries, such as persons, notes, dates, etc. The overview shows which entries are available in an application. When you click on a row in the overview, you select an "Entry". RCU retrieves from the Newton the full entry and displays it on the right side of the screen under the File in: pop-up menu.



Person:
Name:
Honorific: Mr.
First: Steve
Last: Clark
Home: (801)277-2967
Work: 578-6683
Card Type: 1

Mr.

Think of an entry as a "record" in a database. Each entry consists of fields, and each field is identified as a certain type (e.g., the Home and Work phone fields are number types). The entry displays all the fields you have added or will add to that entry.

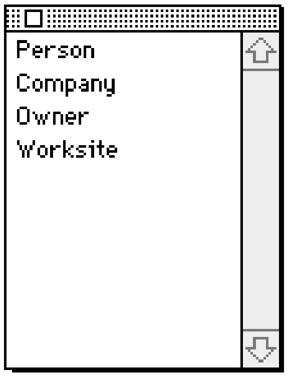
The example above shows an entry (person) for the Names application. The fields added to this entry include Name (with subfields Honorific, First, and Last), Home phone number, Work phone number, and Card Type.

Adding Entries

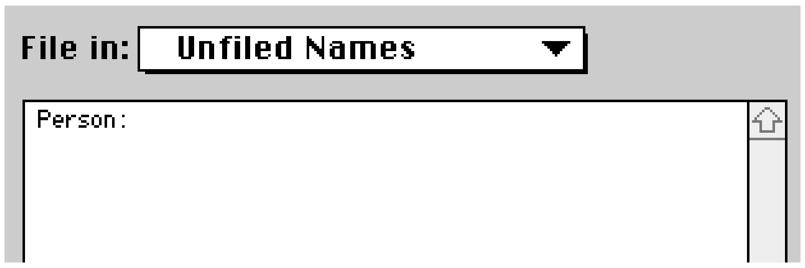
When you view a Newton application through RCU, existing entries display in the overview. You can add new entries to the Newton application through RCU by selecting Add [entry]… from the Action menu. The name of this menu item is customized for the application that is open. For example, if you are looking at Names, then you select Add Name… from the Action menu to add a new entry (name).

Stationery Palette

When you add an entry, RCU opens a palette listing the types of entries available. This palette of different entry types is called the Stationery. The selections available are specific to the current application. For example, the following stationery appears for the Names application.

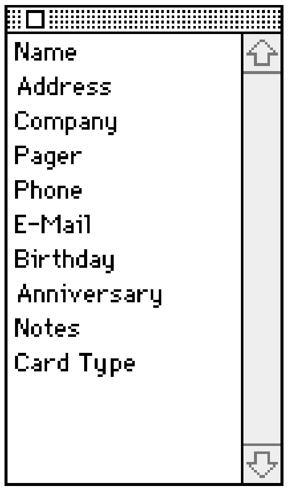


Click on the type of entry you want to add. The palette automatically closes and the new entry appears in the Entry table under the File in: pop-up menu.



Adding Fields

When you first add a new entry, only default fields are added. If it is an existing entry, only those fields that have been added display. You can add more fields by selecting Add Field… from the Action pull-down menu. A palette of fields available for that entry appears. For example, the following Field palette appears when you add a new name (person):



Simply click on a field on the palette to add it to your entry. The field "moves" from the palette to the Entry table and the cursor appears in the Field Editor bar, where you can start typing in the field information. Fields that allow only one occurrence of information (e.g., birthday) don't appear on the palette once you have selected it.

Note and picture fields appear over and hide the Entry table. This allows you to add as much information as necessary and use the return key to separate text.

Compound Fields and Subfields

Certain fields, such as Name, consist of subfields: Honorific, First, and Last. Fields with subfields are called compound fields. If you inadvertently delete a subfield, you can view the palette of subfields available for the compound field by clicking on the compound field.

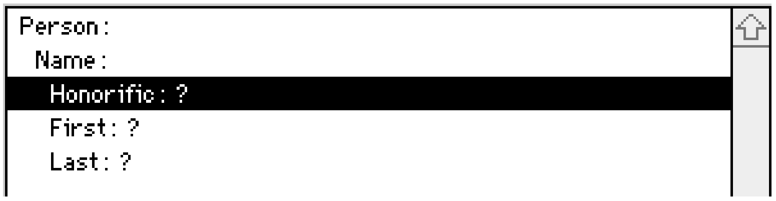
Navigating through Fields

To navigate forward through the fields in an entry, press the TAB key or click on the desired field. To move backward, press SHIFT+TAB. If you tab through a conditional field that has no information in it, pressing tab deletes the field from the Entry table.

***Note*: You MUST press the ENTER key to save changes to the field, NOT the RETURN key. The ENTER key is found on the numeric keypad. Also, pressing ENTER only saves the data into the field, it does NOT save the entry out to the Newton. You must select Save from the Action menu to save the entry to the Newton.**

Conditional Fields

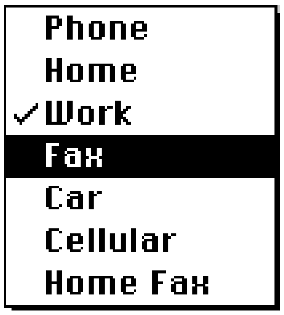
Certain fields are conditional and don't need to be completed. These fields are identified with a ? when first added. For instance, the Name subfields are conditional:


Person:
Name:
Honorific: ?
First: ?
Last: ?

If you tab through a conditional field that hasn't been completed, RCU deletes the field from the entry.

Specifying Class Type

If the field has a class type, such as a phone number, the class pop-up menu is active. For example, you specify the class of a phone number by clicking on the appropriate pop-up menu item.



Editing Fields

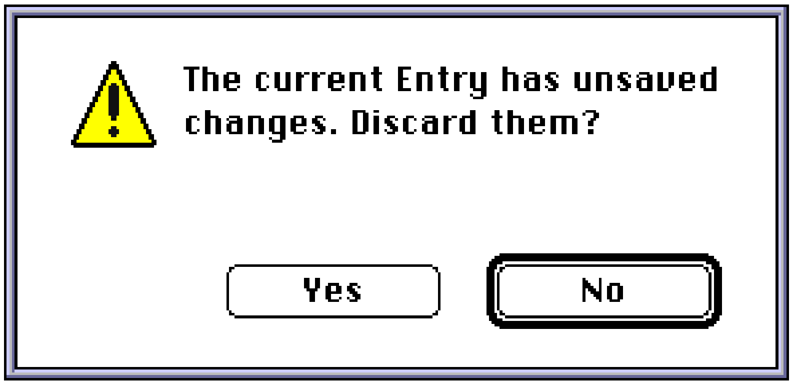
To edit a field in the entry you click on it, or begin tabbing through each row until you arrive at the desired field. Once the field is highlighted, the field editor displays the contents of the field. You add, edit, or delete the field contents by typing within the field editor.

***Note*: You MUST press the ENTER key to save changes to the field, NOT the RETURN key. The ENTER key is found on the numeric keypad. Also, pressing ENTER only saves the data into the field, it does NOT save the entry out to the Newton. You must select Save from the Action menu to save the entry to the Newton.**

Locked Applications, Entries, and Fields

An application is locked if next to its name is the word [Locked]. Certain entries and fields within non-locked applications may be locked as well, as designated by the padlock 🔒. A locked application or field is read-only, which means you can view the contents but not make changes.

Saving Changes to Your Newton

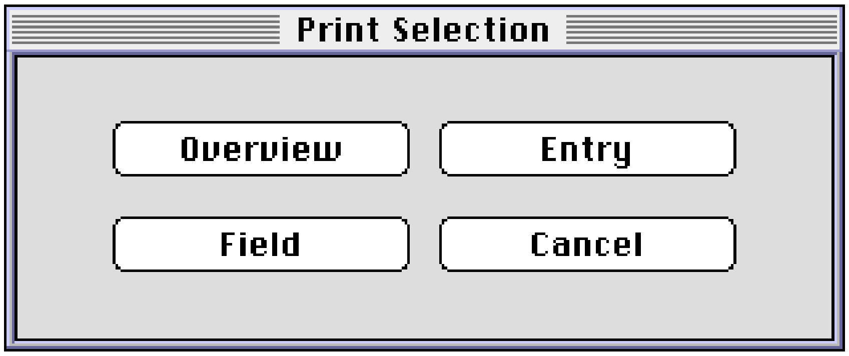

The current Entry has unsaved changes. Discard them?

If you add a new entry, file an entry in a different folder, or make additions and/or changes to fields in an entry, you must save the changes back to the Newton. Unsaved changes are indicated by the appearance of a small floppy disk icon 💾. To save changes, select Save from the Action menu. After saving, the overview reloads the new information.

If you attempt to open a new application or select a different entry before saving changes to the current entry, RCU alerts you with a message. Selecting No (pressing enter) takes you back to the current entry so that you can save your changes.

Printing

You may print either the Overview, the Entry table, or fields (through the field editor), depending on the preference default (see "Setting Preferences" in this guide). Printing the overview only prints the first 200 rows, and prints exactly as shown on the screen. Selecting the field editor option lets you print notes. To print, click on the print icon or select Print… from the File menu. If you select the "Ask me which to print" preference, the Print selection window appears from which you can specify what to print.

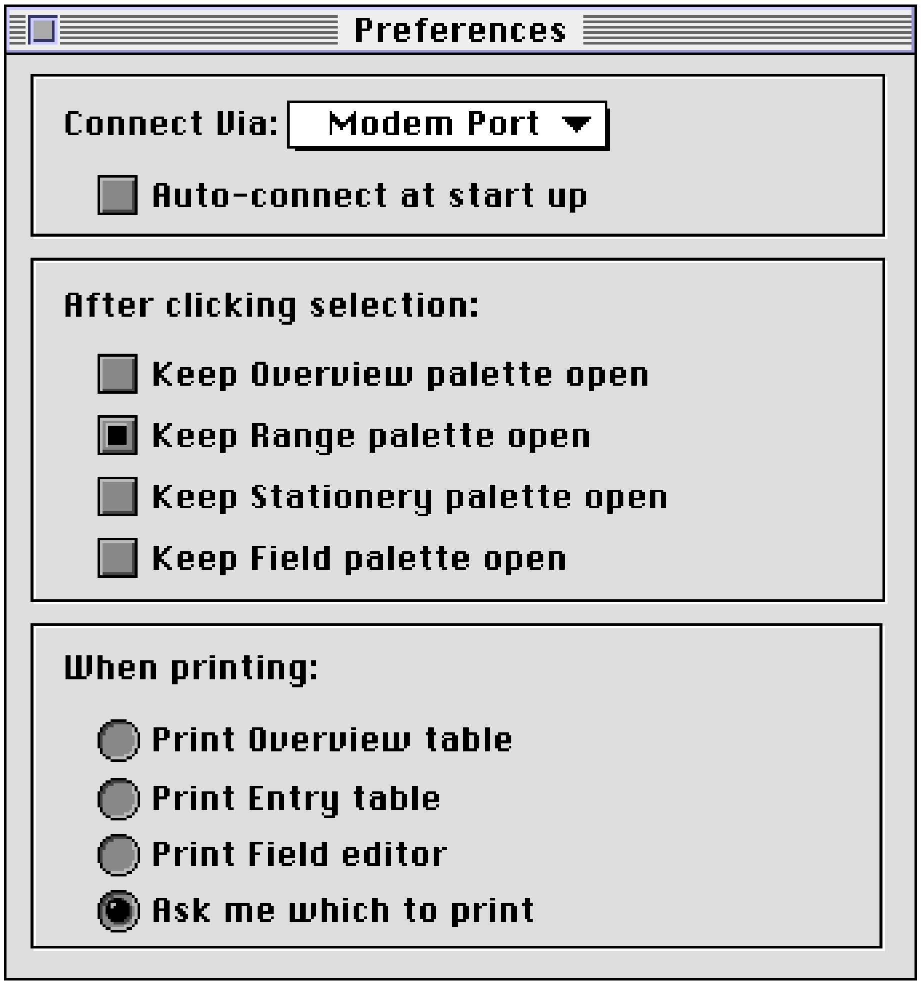


Changing Applications

The Applications menu lists all applications available on your Newton. Use this menu to select a different application. The overview and entry information changes when you select a different application. The title bar on the main window shows the name of the currently selected application.

Setting Preferences

The Preferences… selection under the Edit menu gives you options for the way you use RCU. There are three sections:


Preferences

Connect Via: Modem Port

☐ Auto-connect at start up

After clicking selection:

☐ Keep Overview palette open

[✓ Keep Range palette open]

☐ Keep Stationery palette open

☐ Keep Field palette open

When printing:

○ Print Overview table

○ Print Entry table

○ Print Field editor

[✓ Ask me which to print]

Connection

The first section of the Preferences window lets you specify which port RCU should use when trying to make a connection to the Newton. You can select any available serial port, but the current version of RCU does not support AppleTalk. Check on the Auto-connect at start up button if you want RCU to automatically attempt to connect to the Newton device as soon as it launches. Checking this box only saves you the trouble of clicking Connect on the Macintosh side. You must still press the Connect button the Newton portion of RCU.

Palette Preferences

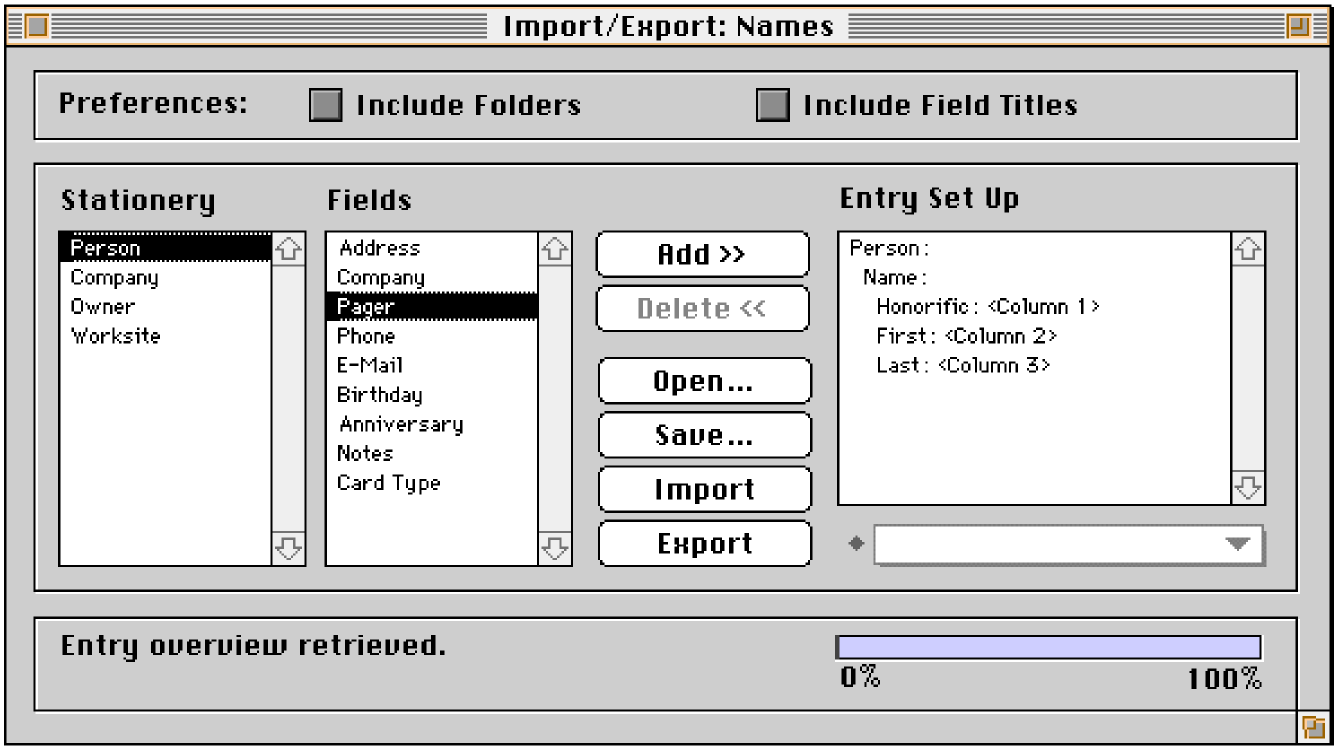
There are four palette windows in RCU. After you click on a selection in a given palette, the palette automatically closes. If you wish to have a palette remain open after clicking your selection, check on the appropriate box. Keeping a palette open might be desirable, for example, in the case of the Fields palette, where you may wish to rapidly add different fields to a Names entry.

Printing

The third section of the Preferences window lets you specify the default for printing in the main window. You may print either the overview, the entry, or field editor (which allows you to print notes). Printing the overview only prints the first 200 rows, and prints exactly as shown on the screen.

Importing and Exporting

RCU lets you transfer entries from or to your Macintosh in tab-delimited text files. In this type of file, each field is separated by a tab. The Import/Export window lets you choose which entries and fields are exported from your Newton. You can also use this window to import tab-delimited text files to your Notes, Names, Calendar and other applications, or any ADF application designed for your Newton.



Person:
  Name:
    Honorific: <Column 1>
    First: <Column 2>
    Last: <Column 3>

Entry overview retrieved.

To open the Import/Export window, select Import/Export… from the File menu. The name of this menu item is based on the current application open. For example, if you have the Names application open, you select Import/Export Names… to open the Import/Export window.

Exporting

The Import/Export window lets you specify which information (i.e., entries and fields) you want to export from your Newton. You add the fields for an entry into the Entry Set Up window. The process is simple. The basic steps are provided here:

1. Select the type of entry you want to export.

For example, in the Names application you can export the following entry types: person, company, owner, or worksite. The list of entries available for the application—the stationery—appears in the far left column.

The list of fields available for the selected entry type appears in the next column. The list of fields changes depending on the entry type selected from the stationery.

2. Select the field(s) you want to export and press Add >>,

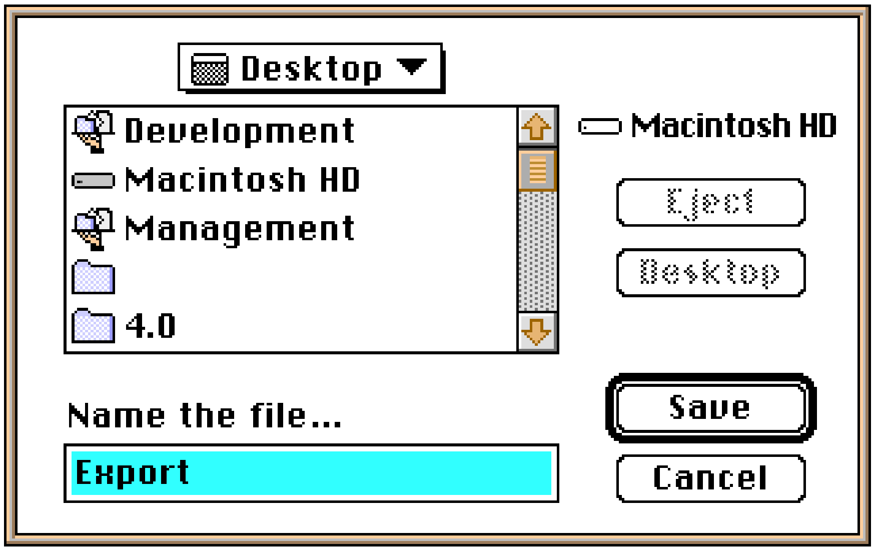
You must add each field one at a time. Double-clicking on a field selects and adds the field at the same time. The entry type (e.g., Person) and selected fields appear in the Entry Set Up window. If a field has subfields (e.g., Name has Honorific, First, and Last), the field and all subfields are added at once. Each field (and/or subfield) is assigned a column number, which designates the order in which the information will be exported. Each column (field) will be separated by a tab in the export document.

3. To remove a field (subfield) from the export document, click on the field (subfield) in the Entry Set Up window and press Delete <<.

The "deleted" field is removed from Entry Set Up and the remaining fields are reassigned column numbers.

4. Click Export to create an export document.

A dialogue box appears in which you can assign a name to the export document and file it in a folder on your Macintosh.



Name the file...

Export

Importing

You can import documents into the current Newton application selected in RCU, provided you have created columns in your import document that correspond to how the fields are ordered in the application.

Note: The order of fields is shown in the Fields palette and is listed in the newton.adf file, which is installed in the ADF Folder when you install RCU on your Macintosh. You can change the order of fields in the newton.adf file. See "Advanced User Information" in this guide for details.

For example, if you want to import a spreadsheet, save the spreadsheet file as a text file and make sure it is laid out in consistent columns.

To import a file into the current application:

1. Select the type of entry you want to import into.

For example, in the Names application you can send the information from the import document into the following entry types: person, company, owner, or worksite. The list of entries available for the application—the stationery—appears in the far left column.

The list of fields available for the selected entry type appears in the next column. The list of fields changes depending on the entry type selected from the stationery.

2. Select the field(s) you are importing information into and press Add >>,

You must add each field one at a time. Double-clicking on a field selects and adds the field at the same time. The entry type (e.g., Person) and selected fields appear in the Entry Set Up column. You can also identify fields with class types (e.g., specify the Phone field as Fax).

If a field has subfields (e.g., Name has Honorific, First, and Last), the field and all subfields are added at once. Each field (and/or subfield) is assigned a column number, which designates the order in which the information will be imported. Each column (field) must be separated by a tab in the import document.

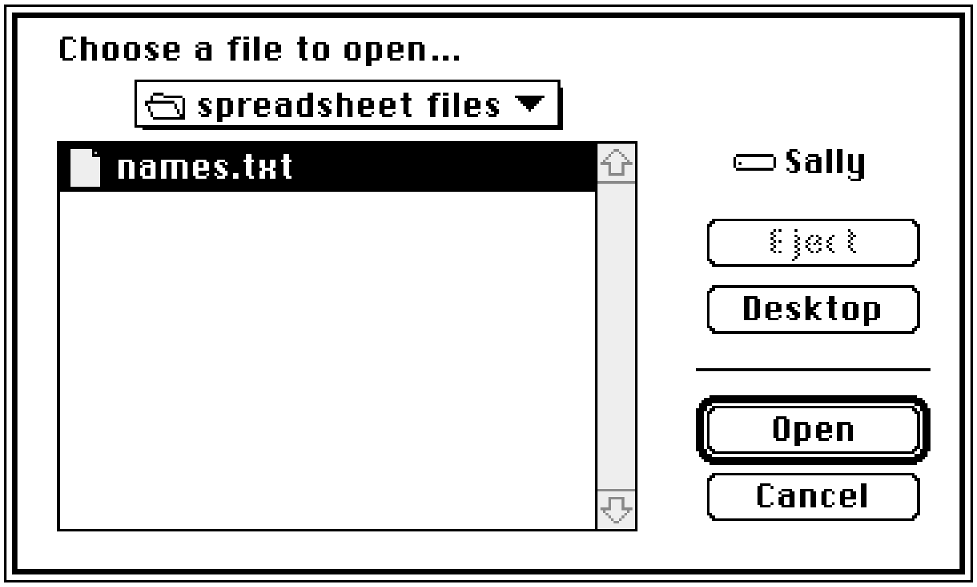
Important: Make sure that your import document corresponds *exactly* to the field arrangement shown in the Entry Setup. Otherwise, the wrong information will be imported into the fields. If the import document has duplicate information for a specific field (e.g., two phone numbers), make sure you add the field to the Entry Setup as many times as necessary. For example, if you are importing a file of names where each name (entry) can have anywhere from one to four phone numbers, make sure that you add the Phone field four times to the Entry Setup.

3. To remove a field (subfield) from the Entry Setup window, click on the field (subfield) in the Entry Set Up and press Delete <<.

The "deleted" field is removed from Entry Set Up and the remaining fields are reassigned column numbers.

4. Click Import to select the import document.

A dialogue box appears in which you can retrieve the import document from the designated folder on your Macintosh.



5. Select the import document and click Open.

Make sure the document has been saved in a tab-delimited text format.

The Import/Export window shows the progress of the import. Each entry is displayed in the Entry Setup.

Exporting and Importing Note Fields

Certain Newton applications (e.g., Notes and Names) have the Note or Notes field, a free-form field wherein you can add notes that can consist of long blocks of text with tabs, returns, etc. This flexible, free-form nature of the Note field creates a special situation in the export/import process because export and import documents are in the form of tab-delimited files, where each entry is a "row" separated by a return, and each field is a "column" separated by a tab.

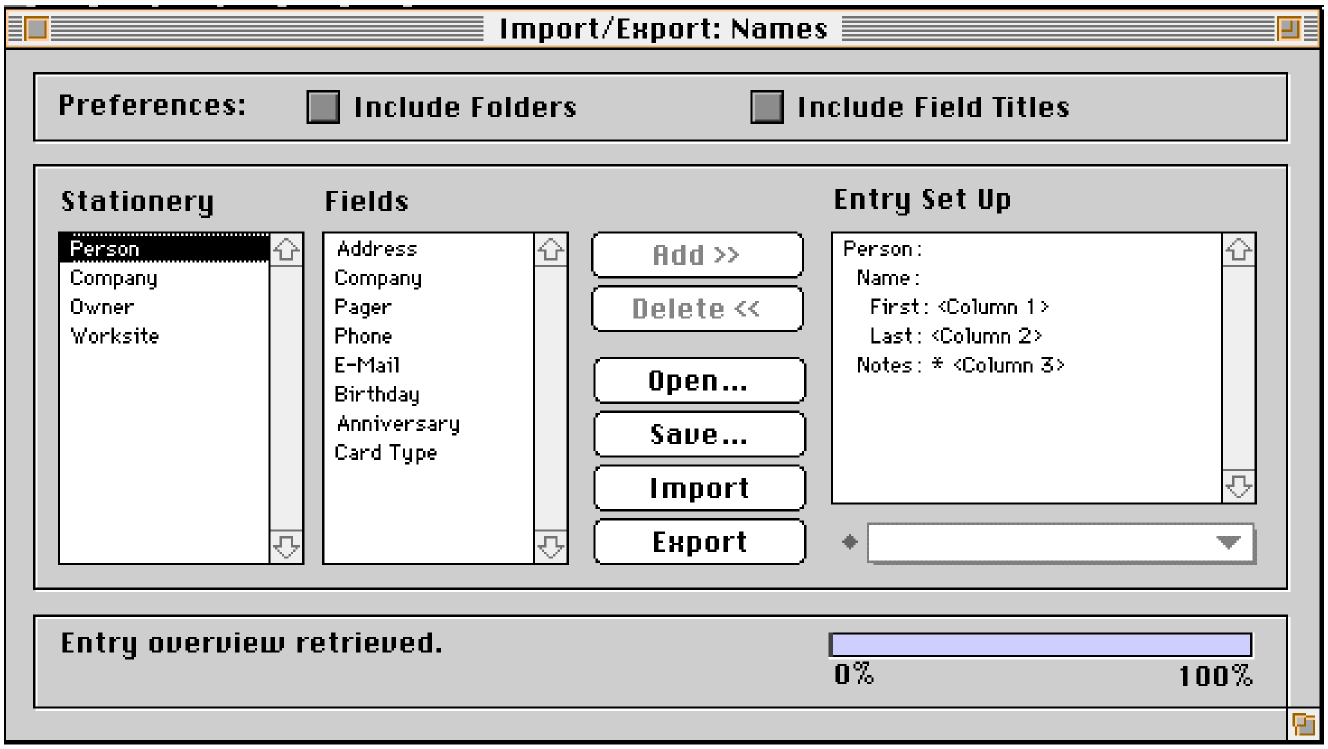
Exporting Notes

When you export entries with the Note field in the Entry Set Up, RCU creates a tab-delimited export document (the "export document") and then exports the Note fields as SEPARATE text documents (the "notes"). Notes are saved in a folder with the same name as your export document, except with a "ƒ" at the end. The name of each document is the title of the note if you are in the Notes application *and* you have included the Title field in your export Entry Set Up or template. This applies to other applications using Note fields as well by the following rule: If there is only 1 other text field in your import/export Entry Set Up or template, the text document takes that text field for its own title. For example, if I am exporting from the Names application, and my Entry

Set Up has Last Name and Notes fields as the only 2 columns, RCU will create a text document for each name that has a note in it and title that document by the Last Name of that entry (i.e. Smith).

If there is more or less than 1 text field, RCU titles the text documents with numbers that refer to the entry row and column of the Notes field in the export document.

For example, if the Note field is column 3 and there are 3 entries (rows 1–3) in the export document, then the notes are named 1-3, 2-3, 3-3. In the export document, the Note field column lists the location (path) and name of the folder where the notes are saved. Here is an example of setting up an export document for the Names application:


Import/Export: Names

Preferences: Include Folders Include Field Titles

Stationery: Person, Company, Owner, Worksite

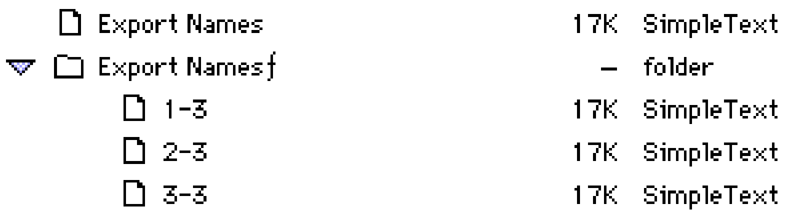
Fields: Address, Company, Pager, Phone, E-Mail, Birthday, Anniversary, Card Type

Entry Set Up:
```
Person:
  Name:
    First: <Column 1>
    Last: <Column 2>
  Notes: * <Column 3>
```

Entry overview retrieved.

In this example, the fields First Name, Last Name, and Notes are exported, with the Notes field as the last column.

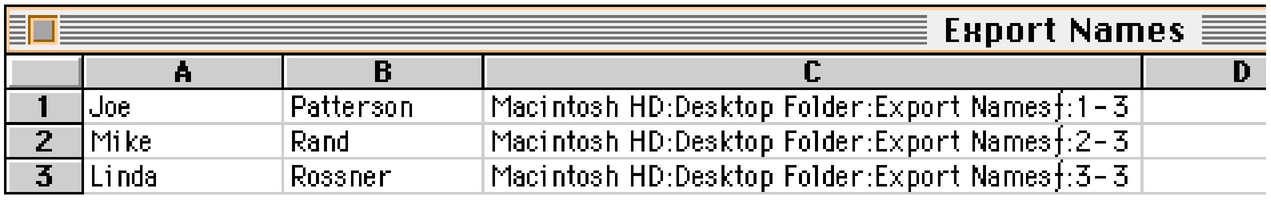
After the export, the export document and notes are saved to the designated folder on your Macintosh. Here is an example of how the export document (named Export Names) and notes might appear in Finder:



Export Names	17K	SimpleText
Export Namesƒ	–	folder
1-3	17K	SimpleText
2-3	17K	SimpleText
3-3	17K	SimpleText

The location and name of each note is referenced in the Note field column (designated as column 3 in this example) of the export document.

Here is an example of how the export document might look when opened into a spreadsheet:


Export Names

	A	B	C	D
1	Joe	Patterson	Macintosh HD:Desktop Folder:Export Namesƒ:1-3	
2	Mike	Rand	Macintosh HD:Desktop Folder:Export Namesƒ:2-3	
3	Linda	Rossner	Macintosh HD:Desktop Folder:Export Namesƒ:3-3	

The column for the Note field shows the location (path) first, then the name of the file. In this example, the note named 1-3 is filed in the Export Names*f* folder, which is in the Desktop folder, which is in the Macintosh HD.

Importing Notes

You import notes using the same principles used in exporting notes. Make sure that the import document has a column for the Note field, and that the location and name of each note text document is referenced. You can choose any name for your note text documents (i.e., you aren't limited to RCU's default naming convention of row-column coordinates). Just make sure that the name of the text document and the folder(s) in which it resides are referenced in the import document.

Note: If you want to bypass the tab-delimited import document and just import the note text documents, press the option key and click on the Import button. This opens a dialog where you can select multiple text files to import. If your import set up/template contains the date and title fields, the text document names will become the Note titles and the document creation date will be the Note creation date. Once imported these Notes will be unfiled, so you will have to go to Unfiled Notes or All Notes to see them in the overview.

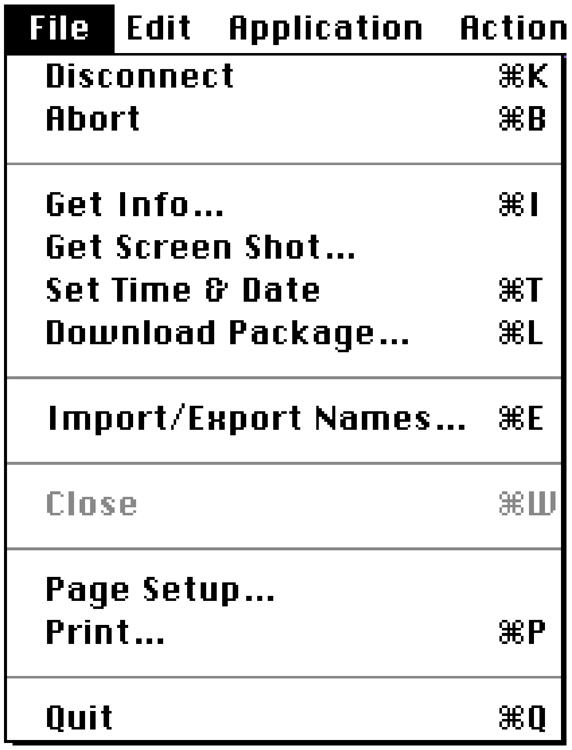
Saving Templates

Since it can be cumbersome to create a new Entry Set Up each time you wish to export or import the same type of data, you can save templates on your hard disk. To do this, just do your Entry Set Up as you normally would, and then press the Save… button in the Import/Export window. RCU then creates a text document detailing your set up and you should title it so that you can remember what application the template is for (i.e., Notes or Names). The next time you wish to import/export with that template, make certain you are in the appropriate application, click the Open… button, and select your template. Do not attempt to open a Calendar template while in the Names application.

Reference Information

This section explains the menu items available for each of the RCU menus and, where applicable, shows the windows accessed through each menu item.

File Menu



Connect/Disconnect

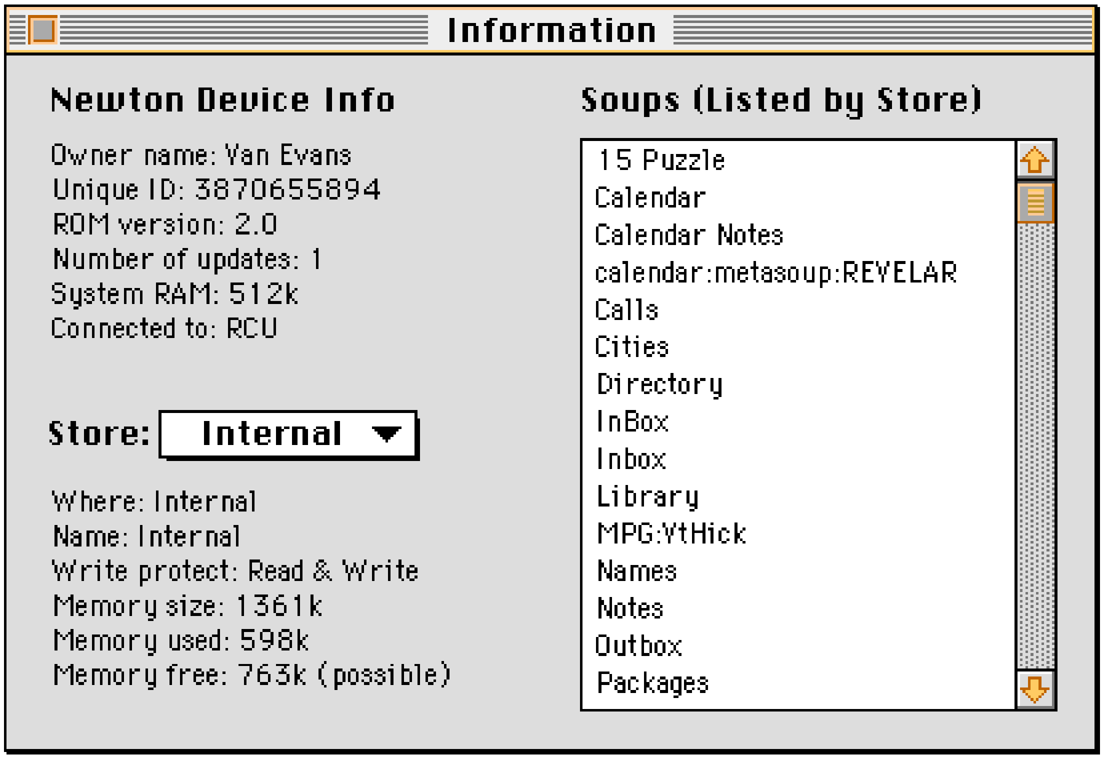
This menu item toggles its name depending upon the state of connection with the Newton.

Abort

This menu item aborts the active command RCU is working on.

Get Info…

This menu item opens the Information window, which gives you storage and other information about your Newton, including information about memory internally and on your PC card, plus a list of soups (data files) found on each store. Select the Store pop-up menu to view information that is internal or on your PC card.



Newton Device Info

Owner name: Van Evans
Unique ID: 3870655894
ROM version: 2.0
Number of updates: 1
System RAM: 512k
Connected to: RCU

Store: Internal

Where: Internal
Name: Internal
Write protect: Read & Write
Memory size: 1361k
Memory used: 598k
Memory free: 763k (possible)

Soups (Listed by Store)

15 Puzzle
Calendar
Calendar Notes
calendar:metasoup:REYELAR
Calls
Cities
Directory
InBox
Inbox
Library
MPG:YtHick
Names
Notes
Outbox
Packages

Get Screen Shot…

This menu item creates a screen image of the current Newton screen. To save the screen image, copy and paste it to any other Macintosh program. If the RCU connect window on the Newton is in the way, just close it. Closing the window does not interrupt connection. This command works even if your rotate your screen.

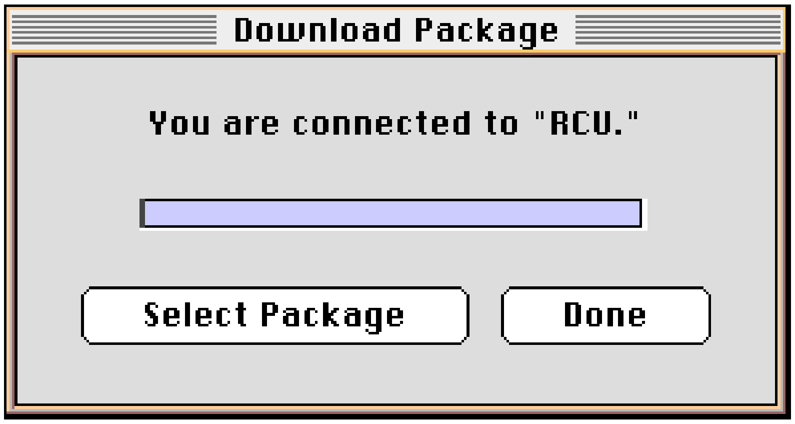
Set Time & Date

This menu item synchronizes the date and time of your Newton with your Macintosh.

Download Package…

This menu item opens the Download Package window, shown on the next page, which permits you to download other Newton software into your Newton device. If you just want to download packages to your Newton, RCU will connect to the Connection application in your Extras drawer.

To download, press the Select Package button. This opens a dialog box asking you to find the package (.pkg file) you wish to download into the Newton device. After selecting the package, click Open. The dialog closes and the thermometer bar in this window monitors the progress of the download.



You are connected to "RCU."

Import/Export XXXX…

RCU lets you transfer entries from or to your desktop in tab-delimited text files. In this type of file, each field is separated by a tab. The Import/Export window lets you choose which entries and fields will be exported from your Newton. You can also use this window to import tab-delimited text files to the built-in Newton applications. See "Importing and Exporting" in this guide for more about this menu item.

Close

This menu item closes the active window as applicable.

Page Setup…

This menu item allows you to specify basic characteristics of your printed pages, such as size, page orientations, etc.

Print…

This menu item prints the current overview or the current entry, depending upon your preference (as determined by the Preferences… menu item under the Edit menu).

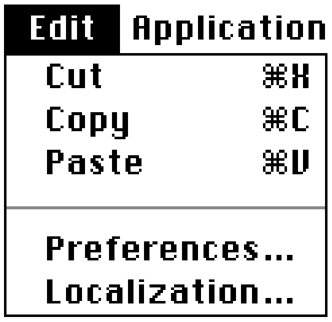
Quit

This menu item quits the RCU application on your Macintosh, and automatically shuts down the connection (if there is one) to the Newton device. No warning is given to save changes.

Edit Menu

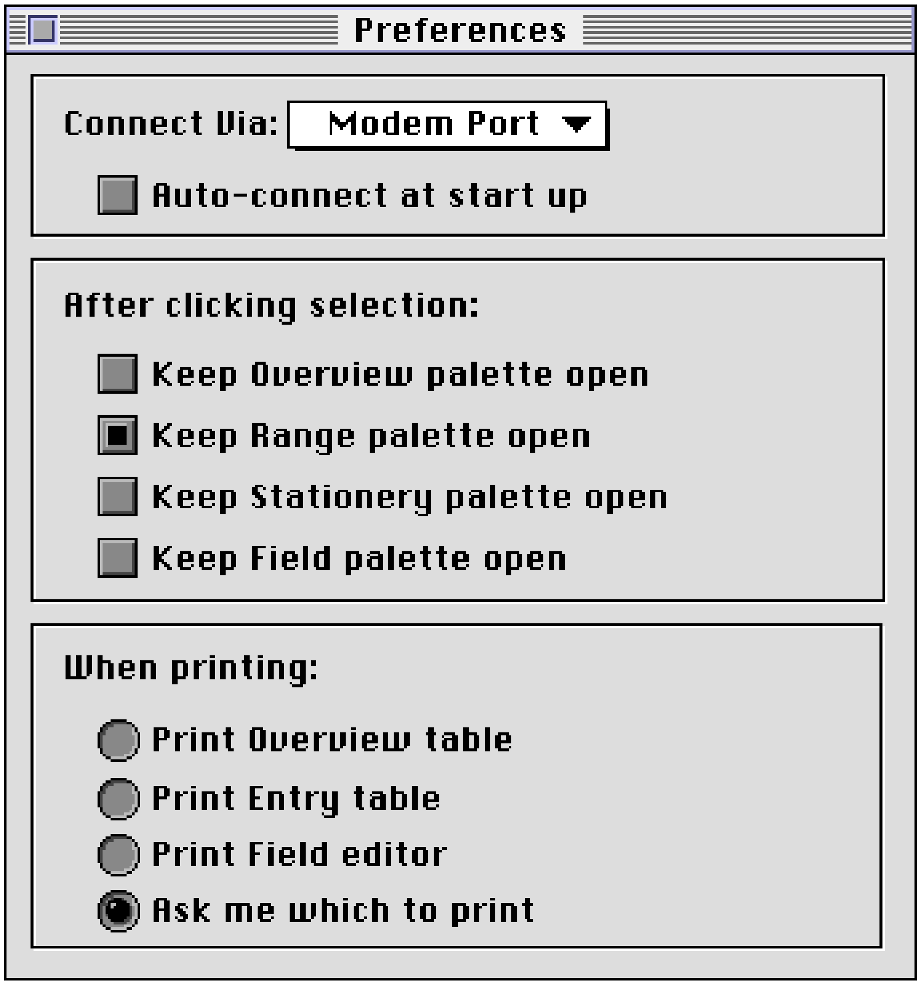
Cut, Copy, and Paste

These items perform their respective functions on all of the editable fields in RCU. You cannot use these items on entries as a single, whole item.



Preferences…

This menu item opens the Preferences window, which gives you options for the way you use RCU. See "Setting Up Preferences" in this guide for details on using this window.



Preferences

Connect Via: Modem Port

☐ Auto-connect at start up

After clicking selection:

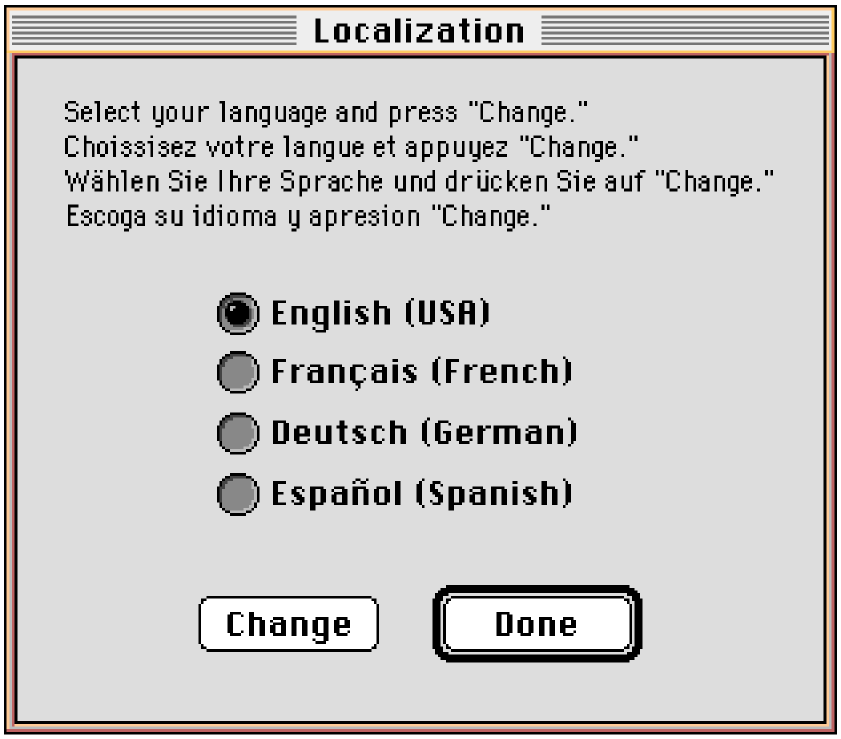
☐ Keep Overview palette open
☑ Keep Range palette open
☐ Keep Stationery palette open
☐ Keep Field palette open

When printing:

○ Print Overview table
○ Print Entry table
○ Print Field editor
● Ask me which to print

Localization…

This menu item opens the Localization window, which localizes RCU into English, German, French, and Spanish.


Localization

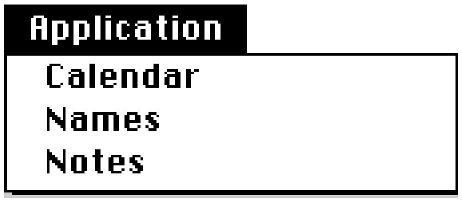
Select your language and press "Change."
Choissisez votre langue et appuyez "Change."
Wählen Sie Ihre Sprache und drücken Sie auf "Change."
Escoga su idioma y apresion "Change."

[✓ English (USA)]
Français (French)
Deutsch (German)
Español (Spanish)

To localize, simply select the desired language and press the Change button.

Note: You must be disconnected to localize.

Application Menu

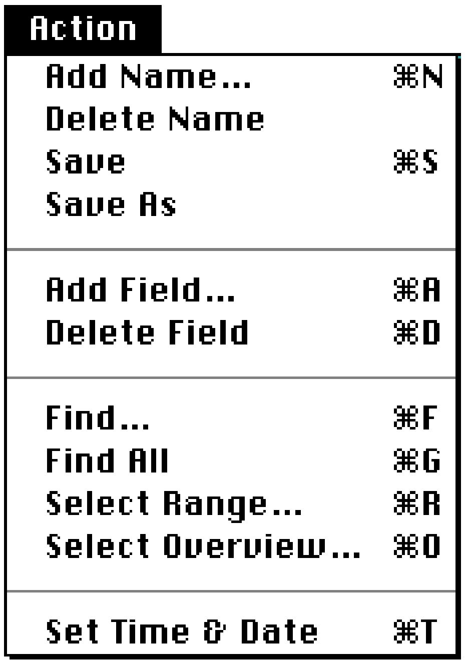


This menu is built each time you connect to RCU. We developed RCU to be flexible and extensible. This means that besides the built in applications, other applications on your Newton can display their data through RCU. This is done through the creation of an ADF (Application Definition Format) file. If you are interested in developing your own ADF file, see the "ADF File Format" documentation at the end of this guide. When you disconnect, the Application menu is disabled and the menu items are removed.

Action Menu

Add XXXX...

This menu item is built each time you change the application. Selecting the item opens the Stationery palette, which contains a list of entry types for a given application. For example, in the Notes application, stationery items include Notes, Checklist Notes, and Outline Notes. After the palette opens, click on the Stationery type you want to add and a new and blank entry is created.



Delete XXXX...

This menu item is built each time you change the application. Selecting the item deletes the current entry.

Note: While the overview will reflect that the entry has been deleted, the entry itself will still be displayed in the Entry table. You can "undo" the delete operation at this point by selecting Save under the Action menu. Otherwise, when you move to another entry, press the "Yes" button when warned about discarding the entry. Once you have pressed "Yes," there is no way to recover the deleted entry.

Save

This menu item saves the current entry to the Newton.

Save As…

This menu item duplicates the current entry and rebuilds the overview. After the entry is duplicated, you could click on it, modify it, and then Save it back out.

Add Field...

Selecting this item opens the Field palette. The Field palette contains a list of possible Field items for a given application. For example, in the Names application, Field items include Addresses, Last Names, and Phone numbers. After the palette opens, click on the Field type you want to add and a new and blank Field is created in the current entry.

Delete Field

Selecting this item deletes the selected field. You can undo this operation by not saving the entry.

Find

This menu item opens the Find window, where you can do a search for entries that match your criterion. This only finds entries in the currently selected application and in the current overview.

Find All

This menu item loads into the overview all of the entries for the selected folder and the specified range.

Select Range...

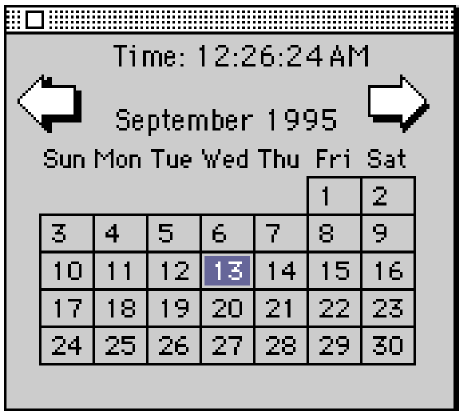
Selecting this item opens the Range palette. The Range palette contains a list of possible Ranges for a given application. For example, in the Calendar application, Range items include This Week, This Month, etc., and defaults to the current date. After the palette opens, click on the Range of entries you want to display in the overview.

Select Overview..

This menu item opens the Overview palette, allowing you to select a different set of columns to show in the overview. For the more adventurous user or developer, RCU is designed so that you can create your own overview.

Calendar

The Calendar window is normally hidden, since it cannot be localized (It is always in English). Open the Calendar window by pressing the Z key while holding down the option key.



Press the arrows to move to the previous or next month in the calendar. If you hold down the option key while pressing the arrows you move to the previous or next year for the shown month.

Advanced User Information

ADF File Format

The Newton 2.0 operating system provides new possibilities for extending the applications that run on the Newton platform. This makes Newton devices very powerful for both data acquisition and user customization.

This extensibility makes it very difficult to write a generalized desktop connectivity application because each Newton device could have different types of extensions, even extensions to“ built-in” applications. Because of the inability to“ hard code” application definitions into an application, Revelar and Network Multimedia decided to externalize this information in the form of an ADF file, or Application Definition Format file.

The format of this file is simple enough that non-programmers can understand it, yet powerful enough to describe almost any Newton data structure. It focuses on defining applications, soups, entries, and extensions.

Overview of the ADF File Format

You use the ADF file format to define entries in soups that exist on Newton devices. It describes most formats that can exist, including all built-in applications. It is also modular, meaning that each block of information builds on previous information, even if split across multiple files. This means that a single file can be used to describe the standard Newton application formats and other files can be included by third parties, each describing either a new application or an extension to a previously defined application.

Another powerful use of the ADF file is localization. Since most information presented to the user in RCU comes from the ADF file, changing the language of the strings in the ADF file has the effect of localizing the RCU application.

Because ADF files can refer to information in previously read files, the order in which ADF files are read is critical. For the RCU application, the first ADF file read is the newton.adf file. If this file is not found, then a default version, coded into the program, is used. All other files in the folder are read in alphabetical order (of course the newton.adf file is skipped during this phase). All ADF files are read from a folder called ADF Folder that must exist in the same directory as the RCU application. If the folder does not exist then only the built-in newton.adf file is used, giving access to only the built-in features of the Newton 2.0 applications.

General Format

ADF files are stored as text files. The standard suffix for these files is “.adf”. The files can be created with any standard text editor and are completely cross platform. Comments can be included in ADF files using the C++ standard, meaning that comments begin with two forward slash characters (//) and extend to the end of the line. In several places the syntax allows for a list of the same object. In this case an ellipsis (...) is used. Throughout this document certain items are referred to, including the following:

Item	Syntax	Example
string	" <any chars but return> "	"Names"
symbol	Alphabetic, then alphanumerics	| <any chars but return> |
		Slot9
		|a-b.5|
number	List of digits	136
path	Alphabetic, then alphanumerics	| <chars> .<chars> ... |
		Slot9
		|name.first|

There are seven top level objects that appear in ADF files: application, soup, overview, entry, class, range and extension. Each of these blocks is described in the following sections. Blocks cannot be split across files, but that blocks in one file can refer to blocks in a previously read file.

Application Syntax

application string called string with options { string ... }

Strings

The first string is the application symbol, a unique identifier for every Newton application. It must be obtained either from the author of the application or by inspection of the Newton data structures.

The second string is the user name for the application or the text that appears in RCU.

Options In Braces

localfolders. The application doesn’t use global folders.

nofolders. The application doesn’t support filing.

readonly. Application data can be viewed but not changed. This option allows export, but not import.

Soup Syntax

soup string called string of string range string with options { string ... }
metasoup string called string of string range string with options { string ... }

This block describes a soup. A single application may have more that one soup. Each application/soup pair appears in the Application menu of RCU.

Strings

The first string is the soup name on the Newton. It can be obtained either from the author of the application that owns the soup or by inspection of the Newton data structures.

The second string is the user name for the soup, or the text that appears throughout RCU.

The third string identifies the application that the soup belongs to. The string must be equal to the application symbol of an application block that has already been encountered.

The range information is optional. If given, the string refers to a range definition that will be provided later on in the file.

Metasoups

The second syntax defines a special object called a metasoup. A metasoup defines an object that is not a soup, but that can be treated like a soup by RCU. Metasoups represent extensions to the RCU package on the Newton that provide an API that RCU talks to by way of frames.

The most common example of a metasoup is the built-in Dates application on the Newton—the format of the entries in the application soups are either undocumented or too complex to map directly into RCU (repeating meetings, etc.). If developers are interested in how to extend RCU in this manner, they should contact technical support.

Options In Braces

readonly. Soup data can be viewed but not changed. This option allows export, but no import.

nofolders. The soup does not use folders, even if it belongs to an application that does support folders.

Overview Syntax

overview string called string of string queries string contains { column ... }
where column has the following syntax:
column string width number is path

This block defines an overview for a soup. Information in the overview window of RCU allows the user to view arbitrary information in a tabular format.

Strings

The first string declares a symbol for the overview. It does not correspond to information on the Newton. It must be unique among all overview blocks in all ADF files.

The second string is the user name for the overview, which appears in the Overview palette in RCU.

The third string identifies the soup for which the overview is valid, and is equal to the name of the soup (not the user name).

The fourth string identifies the slot that is indexed for the query for the overview. Normally it is the indexed slot for the soup for which the overview is valid.

Columns

The list of column information defines the columns that appear in the overview. The columns appear left to right in RCU, in the same order that they appear in the list. Each column line defines the information for a column of the table.

The first string defines the user name of the column—the text appears as the header of the column. The number defines the width (in pixels) of the column. The path declares the path to the slot of the entry that appears in the column.

Note: Advanced users can edit the newton.adf file to create custom overviews (i.e., change the column names or display different information). See "Creating Custom Overviews" in this section for instructions. See "Understanding Overviews" in this guide for an introduction to overviews.

Entry Syntax

class string called string of string with options { string ... } contains { slot ... }
where slot is one of the following:
symbol : type called string class string default string with options { string ... }
symbol : set called string first symbol with options { string ... } [array contents ...]
symbol : entry called string with options { string ... } contains { slot ... }

```
symbol : class string called string with options { string ... } contains { slot ... }
symbol := symbol string
symbol := bounds { number number number number }
symbol := value number
symbol := nil
symbol := format string { path ... }
```

The Entry Syntax is the most complicated and powerful block in the ADF syntax. It defines the contents of an entry of a soup in terms of the slots and types of objects that exist in the entry.

Strings

The first string is the class of the entry. It must be obtained from the author of the application that owns the soup the entry appears in, or from the author of the extension. It is the value of the class slot for the entry.

The second string is the user name for this type of entry and it appears in several places in RCU.

The third string defines the name of the soup that the entry belongs in (a reference to a soup also references an application because soups are attached to applications).

Optional strings are listed below the third string. The slot list defines the contents of the entry. Each of these lines begin with a symbol and the rest depends on the type of information stored. Because of the complex formats involved, each slot type and the options associated with it are described in the following sections.

Simple Slot

```
symbol :  type called string class string default string with options { string ... }
```

This syntax defines a simple slot with a given type:

boolean. A true/false value.

integer. A number, no floating point.

real. A floating point value.

string. A single line of text, no returns allowed.

text. Multiple lines of text, no graphics.

note. A Newton Note object, both text and graphics.

date. A date.

time. A time.

datetime. A single value storing both date and time.

The first string defines the user name of this slot. It appears in the palette when the user chooses to add a slot in RCU. If the value has a class, then the class string is given. The class string in this case refers to a class block, described below, and not to a Newton class (although the class block does define the ultimate class the object will have). If a default value should be given to the object whenever it is added to an entry then it is listed.

Options strings can follow, and come from the following list, which applies for slots and for entries.

hidden. The value is invisible.

suffixfromclass. When the value is displayed, a suffix will be added based on the class the object has.

titlefromclass. When the value is displayed, the title of the object will be based on the class the object has. It will override the user name declared in the ADF file.

required. The slot must be present in order for the entry to be saved to the Newton. Note that this option implies the **automatic** option.

automatic. The slot will be added automatically whenever an entry is created.

readonly. The value can only be viewed, not edited. This will allow export, but not import of this slot.

displayifempty. The slot should be displayed even if it contains an empty value. The definition of Empty changes for each type. As an example, an empty string value would normally not be shown in RCU. This option overrides this behavior.

firstclasswild. The first class defined for this slot is "wild", meaning that if the slot has a class other than those defined for it the system may use the first class definition. This is useful for things like phone number definitions, where the first class is generic.

outlinenote. Used on arrays slots that define the contents of "Outline" notes on the Newton. Used as a flag for import/export that the contents of the array should be translated to or from a textual format.

checklistnote. Used on arrays slots that define the contents of "Checklist" notes on the Newton. Used as a flag for import/export that the contents of the array should be translated to or from a textual format.

Array Slot

symbol : set called string first symbol with options { string ... } [array contents ...]

This syntax defines an set. The first string gives the user name of the set. The first value, if given, tells the system that the first setelement is stored in the root object and not in the array. It may either be a slot name, in which case the array contents should be single values (like strings), or it can be empty, in which case the array contents should be a frame. The way to specify an empty symbol is with a set of vertical bars (||). The options list is the same as for simple slots.

The set contents can either be another array definition, a frame definition, or a simple slot definition, but the symbol and colon are left off. See "Example ADF File" in this guide for clarification.

Note that sets are slightly different from arrays, which are not yet supported. In a set each element is interchangable with any other element, meaning there is no ordering to a set.

Entry Slot

symbol : entry called string with options { string ... } contains { slot ... }

This syntax defines a subframe of an entry or frame. The first string defines the user name of the slot. The options are the same as for a simple slot. The slot list is equivalent to the slot list for the entry block. Entry definitions are used frequently as array elements, where the array elements are unclassed frames of information.

Class Slot

symbol : class string called string with options { string ... } contains { slot ... }

Class slots allow a single slot to have multiple definitions, based on the class given the frame. They are equivalent to entry slots, but allow the same symbol to have multiple definitions. If a slot did have multiple definitions it would appear twice in the slot list, each time with a different class slot definition. Other than that difference, the values are the same as for an entry slot.

Computed Slots

symbol := symbol string
symbol := bounds { number number number number }
symbol := value number
symbol := nil

These definitions allow for a slot to take a constant value, and force the slot to be hidden. The first definition defines a symbol value to be stored in the slot. The second creates a bounds rectangle. The numbers are in the order of top, left, bottom, and right. The third definition allows for numeric values and the last creates a slot with a nil value.

Format Slots

symbol := format string { path ... }

This definition allows for computed fields—fields that depend on the values of other slots. The current syntax only allows for text slots to be used. The string gives the format used to compute the value and has the syntax of the format string of the sprintf function in the C language. The path list gives paths to all slots used in the format. The following example computes the “sortOn” value in the “ Names” soup:

```
sortOn := format “%s %s” { name.last name.first }
```

Class Syntax

field class string symbol is string string

This block defines class information for a slot value. It is referred to by the class keyword in the slot definition. The value of that keyword must match the first string. Multiple class blocks may share the same class name (they usually will).

The class name does not have any significance on the Newton—it only serves to identify objects that can have these classes. The symbol given is the class that the object will have when this particular class is chosen, and the last two strings given the user name (or title value) for the class and the suffix value of the class.

For examples of the use of this block see the slot definition for a phone number in the Names soup (see “ Example ADF File”). Note that class blocks do not need to appear before they are referenced in an ADF file. Class blocks with the same name can appear in different files, they define new class possibilities.

Range Syntax

range string called string is string

This block defines information about ranges. The first string is the name for a set of related ranges and cooresponds to the name used in the soup definition to refer to the range. The second string is the user name for the range as presented in the range pallette. The third string is the range defintion.

The last string contains the definition of the range. Ranges define four pieces of information: Start closure, start element, last element and last closure. The start and last elements can either be thought of as part of the range or not. If an element is part of the set then it is "closed", otherwise it is "open". Examples of the types of ranges can be found in the same ADF file for the Names, Notes and Calendar applications. If an element is closed then a bracket is used to begin or end the string ('[' or ']'). If an element is open then a parenthesis is used to begin or end the string ('(' or ')'). The characters '...' (three periods with no spaces) must appear between the start and last element defintions.

There are four different types of range elements. Range elements may also be empty (not given), in which case the selection extens to the beginning or the end of the soup. Because of this the string "[...]" would be a global range, including in it every entry in the soup.

Date Range Elements

The first type of range element is a date. A date may be specified as a fixed date or a calculated date. Fixed dates are given as 'mm/dd/yy' or 'mm/dd/yyyy'. Computed dates are given as a keyword, optionally followed by a '+' or '-' and an offset. The keywords for computed dates are: **daystart**, **weekstart**, **monthstart** and **yearstart**. Each of these keywords refers to midnight at the start of the unit (daystart means midnight of the current day). Offsets are numbers followed by a suffix that gives the unit: **h** for hour, **d** for day, **w** for week or **y** for year.

Number Range Elements

Number range elements are given by putting a number for the element.

String Range Elements

String elements are given as strings, in double quote marks. **Note** that since range elements themselves appear in a string that these string range elements must have their double quote marks escaped by putting a '\' character in front of them. See the Names range definition for an example.

Extension Syntax

```
extension of string of string { slot ... }
```

Extension blocks allow extending the definition of an entry definition in another file, allowing third parties to ship small definition files that define just their extensions to other applications without having to modify the definition of the entry definition.

Strings

The first string references the class of the entry, while the second string is the name of the soup the entry belongs to. The slot definition list is equivalent to that used in the original definition of the entry.

Example ADF File

```
// Notes Application
application "paperroll" called "Notes"

// Notes Soup
soup "Notes" called "Notes" of "paperroll" range "optionalDateRange"

overview "notes:date" called "By Date" of "Notes" queries "timestamp" contains
        {
        column "Date" width 155 is timestamp
        column "Note Title" width 200 is title
        }

class "paperroll" called "Note" of "Notes" contains
        {
```

```
	data: note called "Note" with options { "required" "automatic" }
	title: string called "Title"
	timestamp: datetime called "Date" default ""
	viewstationary := symbol "paperroll"
	height := value 200
	}

class "checkList" called "Check List" of "Notes" contains
	{
	timestamp: datetime called "Date" default ""
	title: string called "Title"
	topics: array called "Topics" with options { "checklistnote" "required" } { entry called "Topic"
contains
			{
			mtgdone: boolean called "Checked" default "O"
			hideCount: integer called "Hide Count" default "0"
			level: integer called "Level" default "1"
			text: text called "Text" with options { "automatic" }
			viewbounds := bounds { 0 0 200 30 }
			}
		}
	viewstationary := symbol "paperroll"
	height := value 200
	data := nil
	}

class "list" called "Outline" of "Notes" contains
	{
	timestamp: datetime called "Date" default ""
	title: string called "Title"
	topics: array called "Topics" with options { "outlinenote" "required" } { entry called "Topic"
contains
			{
			hideCount: integer called "Hide Count" default "0"
			level: integer called "Level" default "1"
			text: text called "Text" with options { "automatic" }
			viewbounds := bounds { 0 0 200 30 }
			}
```

```
                }
        viewstationary := symbol "paperroll"
        height := value 200
        data := nil
        }


// Names Application
application "cardfile" called "Names"

soup "Names" called "Names" of "cardfile" range "textRange"

overview "names:phones" called "Phone List" of { "cardfile" "Names" } queries "sortOn" contains
        {
        column "Name" width 150 is sortOn
        column "Phone" width 117 is [ phones, 0 ]
        }

overview "names:email" called "E-Mail List" of { "cardfile" "Names" } queries "sortOn" contains
        {
        column "Name" width 150 is sortOn
        column "E-Mail" width 117 is email
        }

overview "names:company" called "Company List" of { "cardfile" "Names" } queries "sortOn" contains
        {
        column "Name" width 150 is sortOn
        column "Company" width 117 is company
        }

overview "names:bday" called "Birthday List" of { "cardfile" "Names" } queries "sortOn" contains
        {
        column "Name" width 150 is sortOn
        column "Birthday" width 117 is bday
        }

class "Person" called "Person" of { "cardfile" "Names" } contains
```

```
	{
	sortOn := format "%s %s" { name.last name.first }
	name: entry called "Name" with options { "required" } contains
		{
		honorific: string called "Honorific" with options { "automatic" }
		first: string called "First" with options { "automatic" }
		last: string called "Last" with options { "automatic" }
		}

	addresses: set called "Address" firststoredin ll { entry called "Address" contains
			{
			address: string called "Street"
			address2: string called "Street"
			city: string called "City"
			region: string called "Region/State"
			postal_code: string called "Postal Code"
			country: string called "Country"
			}
		}

	companies: set called "Company" firststoredin ll { entry called "Company" contains
			{
			company: string called "Name"
			title: string called "Title"
			}
		}

	pagers: set called "Pager" firststoredin ll { entry called "Pager" contains
			{
			pagerNum: string called "Pager" class "pagerClasses" with options {
"titlefromclass" "firstclasswild" }
			pagerPIN: string called "Pager PIN"
			}
		}

	phones: set called "Phone" { string called "Phone" class "phoneClasses" with options {
"titlefromclass" "firstclasswild" } }
	emailAddrs: set called "E-Mail" firststoredin ll { entry called "E-Mail" contains
```

```
                {
                email: string called "EMail" class "mailClasses" with options { "titlefromclass"
"firstclasswild" }
                }
            }
    bday: date called "Birthday"
    anniversary: date called "Anniversary"
    notes: note called "Notes"
    cardType: integer called "Card Type"
    }


class "company" called "Company" of { "cardfile" "Names" } contains
    {
    sortOn := format "%s" { company }
    company: string called "Company" with options { "required" }
    names: set called "Name" { entry called "Name" contains
                {
                honorific: string called "Honorific"
                first: string called "First Name"
                last: string called "Last Name"
                affiliation: string called "Title"
                }
            }

    addresses: set called "Address" firststoredin II { entry called "Address" contains
                {
                address: string called "Street"
                address2: string called "Street"
                city: string called "City"
                region: string called "Region"
                postal_code: string called "Postal Code"
                country: string called "Country"
                }
            }

    phones: set called "Phone" { string called "Phone" class "phoneClasses" with options {
"titlefromclass" "firstclasswild" } }
```

```
	emailAddrs: set called "E-Mail" firststoredin email { string called "EMail" class "mailClasses" with
options { "titlefromclass" "firstclasswild" } }
	notes: note called "Notes"
	cardType: integer called "Card Type"
	}



class "owner" called "Owner" of { "cardfile" "Names" } with options { "nofolders" } contains
	{
	labels := symbol "_ownerNames"
	sortOn := format "%s %s" { name.last name.first }
	name: entry called "Name" with options { "required" } contains
		{
		honorific: string called "Honorific" with options { "automatic" }
		first: string called "First" with options { "automatic" }
		last: string called "Last" with options { "automatic" }
		}

	addresses: set called "Address" firststoredin ll { entry called "Address" contains
			{
			address: string called "Street"
			address2: string called "Street"
			city: string called "City"
			region: string called "Region/State"
			postal_code: string called "Postal Code"
			country: string called "Country"
			}
		}

	companies: set called "Company" firststoredin ll { entry called "Company" contains
			{
			company: string called "Name"
			title: string called "Title"
			}
		}

	pagers: set called "Pager" firststoredin ll { entry called "Pager" contains
			{
```

```
                    pagerNum: string called "Pager" class "pagerClasses" with options {
"titlefromclass" "firstclasswild" }
                    pagerPIN: string called "Pager PIN"
                    }
            }

        phones: set called "Phone" { string called "Phone" class "phoneClasses" with options {
"titlefromclass" "firstclasswild" } }
        emailAddrs: set called "E-Mail" firststoredin ll { entry called "E-Mail" contains
                    {
                    email: string called "EMail" class "mailClasses" with options { "titlefromclass"
"firstclasswild" }
                    emailpassword: string called "Password"
                    }
            }

        owner: entry called "Owner Information" contains
                    {
                    bankAccounts: set called "Bank Account" { entry called "Bank Account" contains
                            {
                            bankAcctNum: string called "Acct #"
                            bankContactNum: string called "Phone" class "phoneClasses"
with options { "titlefromclass" "firstclasswild" }
                            }
                        }

                    creditCards: set called "Credit Card" { entry called "Credit Card" contains
                            {
                            creditCardNum: string called "Card #"
                            creditCardName: string called "Name" class "creditClasses" with
options { "titlefromclass" "firstclasswild" }
                            creditCardExpDate: date called "Exp. Date"
                            creditCardContactNum: string called "Phone #"
                            }
                    }
            }

        bday: date called "Birthday"
        anniversary: date called "Anniversary"
        notes: note called "Notes"
        cardType: integer called "Card Type"
        }

class "worksite" called "Worksite" of { "cardfile" "Names" } with options { "nofolders" } contains
        {
        labels := symbol "_ownerNames"
        sortOn:= format "%s" { place }
        place: string called "Worksite" with options { "required" }
        areaCode: string called "Area Code"
        dialingPrefix: string called "Dialing Prefix"
        notes: note called "Notes"
        }

field class "creditClasses" |string.card| is "Card" ""
field class "creditClasses" |string.card.phonecard| is "Phone Card" ""
field class "creditClasses" |string.card.creditcard| is "Credit Card" ""
field class "creditClasses" |string.card.phonecard.att| is "AT&T" ""
field class "creditClasses" |string.card.phonecard.mci| is "MCI" ""
field class "creditClasses" |string.card.phonecard.sprint| is "Sprint" ""
field class "creditClasses" |string.card.creditcard.visa| is "VISA" ""
field class "creditClasses" |string.card.creditcard.mastercard| is "MasterCard" ""
field class "creditClasses" |string.card.creditcard.amex| is "AmEx" ""
field class "creditClasses" |string.card.creditcard.discover| is "Discover" ""

field class "pagerClasses" |string.pager| is "Pager" ""
field class "pagerClasses" |string.pager.skytel| is "SkyTel" ""
field class "pagerClasses" |string.pager.mobilecomm| is "MobileComm" ""
field class "pagerClasses" |string.pager.embarc| is "EMBARC" ""

field class "phoneClasses" otherPhone is "Phone" ""
field class "phoneClasses" homePhone is "Home" "H"
field class "phoneClasses" workPhone is "Work" "W"
field class "phoneClasses" faxPhone is "Fax" "F"
field class "phoneClasses" carPhone is "Car" "A"
field class "phoneClasses" mobilePhone is "Cellular" "C"
field class "phoneClasses" homeFaxPhone is "Home Fax" "HF"
field class "mailClasses" lstring.emaill is "Email" ""
field class "mailClasses" lstring.email.internetl is "Internet" ""
field class "mailClasses" lstring.email.aoll is "America Online" ""
field class "mailClasses" lstring.email.compuservel is "CompuServe" ""
field class "mailClasses" lstring.email.mcimaill is "MCI Mail" ""
field class "mailClasses" lstring.email.attmaill is "AT&T Mail" ""
field class "mailClasses" lstring.email.easylinkl is "EasyLink" ""
field class "mailClasses" lstring.email.prodigyl is "Prodigy" ""
field class "mailClasses" lstring.email.geniel is "GEnie" ""
field class "mailClasses" lstring.email.delphil is "Delphi" ""
field class "mailClasses" lstring.email.msnl is "Network" ""
field class "mailClasses" lstring.email.interchangel is "Interchange" ""
field class "mailClasses" lstring.email.radiomaill is "RadioMail" ""


// Calendar Application
// Note that the calendar application is written using metasoups as the
// format is too complex and not really documented.
application "calendar" called "Calendar" with options { "nofolders" }

// Calendar Metasoup
metasoup "calendar:metasoup:REVELAR" called "Meetings" of "calendar" range "dateRange"

overview "meetings:date" called "By Date" of "calendar:metasoup:REVELAR" queries "mtgStartDate"
contains
        {
        column "Date" width 155 is mtgStartDate
        column "Description" width 200 is mtgText
        }

class "meeting" called "Meeting" of "calendar:metasoup:REVELAR" contains
        {
        mtgStartDate: datetime called "Date" with options { "required" }
        mtgText: string called "Title" with options { "required" }
        mtgDuration: integer called "Duration" with options { "required" }
        notes: note called "Notes"
        mtgAlarm: datetime called "Alarm"
        location: string called "Location" with options { "readonly" }
        invitees: set called "Invitees" with options { "readonly" }
                {
                string called "Invitee" with options { "readonly" }
                }
        }

class "repeatingMeeting" called "Repeating Meeting" of "calendar:metasoup:REVELAR" contains
        {
        mtgStartDate: datetime called "Date" with options { "required" }
        mtgText: string called "Title" with options { "required" }
        mtgDuration: integer called "Duration" with options { "required" }
        notes: note called "Notes"
        mtgAlarm: datetime called "Alarm"
        location: string called "Location" with options { "readonly" }
        invitees: set called "Invitees" with options { "readonly" }
                {
                string called "Invitee" with options { "readonly" }
                }
        mtgStopDate: date called "Stop Date"
        repeats: string called "Repeats" class "repeatClasses" with options { "suffixfromclass"
"displayifempty" "required" }
        }

class "CribNote" called "Event" of "calendar:metasoup:REVELAR" contains
        {
        mtgStartDate: date called "Date" with options { "required" }
        mtgText: string called "Title" with options { "required" }
        notes: note called "Notes"
        mtgAlarm: integer called "Days Notice"
        }

class "repeatingCribNote" called "Repeating Event" of "calendar:metasoup:REVELAR" contains
        {
        mtgStartDate: date called "Date" with options { "required" }
        mtgText: string called "Title" with options { "required" }
        notes: note called "Notes"
        mtgStopDate: date called "Stop Date"
        mtgAlarm: integer called "Days Notice"
        mtgStopDate: date called "Stop Date"
        repeats: string called "Repeats" class "repeatClasses" with options { "suffixfromclass"
"displayifempty" "required" }
        }

field class "repeatClasses" |daily| is "Every Day" "Every Day"
field class "repeatClasses" |weekly| is "Every week" "Every week"
field class "repeatClasses" |biweekly| is "Every other week" "Every other week"
field class "repeatClasses" |monthly| is "Every month" "Every month"
field class "repeatClasses" |monthlyByWeek| is "Same week each month" "Same week each month"
field class "repeatClasses" |yearly| is "Every year" "Every year"
field class "repeatClasses" |yearlyByWeek| is "Same week each year" "Same week each year"
field class "repeatClasses" |other| is "Other" ""

// To Do Application
application "todo" called "To Do" with options { "nofolders" "readonly" }

// To Do Soup
soup "To Do List" called "To Do List" of "todo" range "dateRange"

overview "todo:date" called "By Date" of "To Do List" queries "date" contains
        {
        column "Date" width 155 is date
        }

class "todo" called "To Do Item" of "To Do List" with options { "readonly" } contains
        {
        |date|: date called "Date"
        topics: set called "Items" { entry called "To Do" contains
                        {
                        mtgdone: boolean called "Completed"
                        mtgPriority: integer called "Priority"
                        text: text called "Text"
                        }
                }
        }

range "dateRange" called "Today" is "<range> [ daystart ... daystart + 1d )"
range "dateRange" called "This Week" is "<range> [ weekstart ... weekstart + 1w )"
range "dateRange" called "Next Week" is "<range> [ weekstart + 1w ... weekstart + 2w )"
range "dateRange" called "This Month" is "<range> [ monthstart ... monthstart + 1m )"
range "dateRange" called "Next 30 Days" is "<range> [ daystart ... daystart + 30d )"
range "dateRange" called "Next 60 Days" is "<range> [ daystart ... daystart + 60d )"
range "dateRange" called "Previous 30 Days" is "<range> [ daystart - 30d ... daystart )"

range "optionalDateRange" called "All" is "<range> [ ... ]"
range "optionalDateRange" called "Today" is "<range> [ daystart ... daystart + 1d )"
range "optionalDateRange" called "This Week" is "<range> [ weekstart ... weekstart + 1w )"
range "optionalDateRange" called "Next Week" is "<range> [ weekstart + 1w ... weekstart + 2w )"
range "optionalDateRange" called "This Month" is "<range> [ monthstart ... monthstart + 1m )"
range "optionalDateRange" called "Next 30 Days" is "<range> [ daystart ... daystart + 30d )"
range "optionalDateRange" called "Next 60 Days" is "<range> [ daystart ... daystart + 60d )"
range "optionalDateRange" called "Previous 30 Days" is "<range> [ daystart - 30d ... daystart )"

range "textRange" called "All" is "<range> [ ... ]"
range "textRange" called "#..D" is "<range> [ ... \"E\" )"
range "textRange" called "E..H" is "<range> [ \"E\" ... \"I\" )"
range "textRange" called "I..L" is "<range> [ \"I\" ... \"M\" )"
range "textRange" called "M..P" is "<range> [ \"M\" ... \"Q\" )"
range "textRange" called "Q..T" is "<range> [ \"Q\" ... \"U\" )"
range "textRange" called "U..X" is "<range> [ \"U\" ... \"Y\" )"
range "textRange" called "Y..Z" is "<range> [ \"Y\" ... ]"

// Calls Application
application "callapp" called "Calls" with options { "readonly" }

// Calls Soup
soup "Calls" called "Calls" of "callapp"

overview "calls:date" called "By Date" of "Calls" queries "timestamp"
contains
        {
```

```
        column "Date" width 150 is timestamp
        column "Title" width 150 is title
        }

class "calllog" called "Call" of "Calls" contains
        {
        timestamp: datetime called "Date"
        title: string called "Title"
        name: string called "Name"
        phoneNumber: string called "Number"
        notes: note called "History"
        }
```

Customizing RCU

You can create custom overviews in RCU by editing the newton.adf file. You can also change the order of fields, which may be helpful if you want to import an existing document into RCU (e.g., a spreadsheet file of names, addresses, phone numbers, etc.) and you need to change the default order of fields. See "Importing and Exporting" in this guide for information on importing files.

Important: Before making any changes the newton.adf file, save a backup copy of the original file.

Creating Custom Overviews

It is easy to edit the newton.adf file if you want to change the names of overview columns (e.g., edit "Phone" to read as "Telephone") or display new information. All changes are made in the newton.adf file. To create custom overviews:

1. Open the newton.adf file with any text editor (e.g., SimpleText).

2. Find the syntax for the application for which you want to create a custom overview.

The name of each application is preceded by //.

3. Find the string that starts with "overview."

For example, the Names application has two overview strings:

```
overview "names:phones" called "Phone List" of { "cardfile" "Names" } queries "sortOn" contains
        {column "Name" width 150 is sortOn
        column "Phone" width 110 is [ phones, 0 ]}
overview "names:email" called "E-Mail List" of { "cardfile" "Names" } queries "sortOn" contains
```

```
    {column "Name" width 150 is sortOn
    column "E-Mail" width 110 is email}
```

4. Edit the column name(s) or add a new overview.

To make editing changes, type over existing text. For example, you could change " Phone" to "Telephone."

5. Save the newton.adf file as a text file after making changes.

Changing the Field Order

You can edit the newton.adf file to change the order in which fields appear in an entry. This is an option if you are importing an existing document and the information (fields) for each entry are arranged in a different order.

For example, you may want to import a document with 1,000 lines of names (entries), with addresses, phone numbers, etc. (fields) into the Names application. To import this document correctly, the field information must be in the same order as the fields for the Names application. The default order of fields is set in the newton.adf file. Some users find it faster and easier to change the order of fields in the newton.adf file, rather than change the fields in the import document. To change the field order:

1. Open the newton.adf file with any text editor (e.g., SimpleText).

2. Find the syntax for the application for which you want to create a custom overview.

The name of each application is preceded by //.

3. Find the type of entry, called the class, that you want to edit fields in.

The string for each type of entry begins with "class." Fields are listed under each class.

For example, the class string and corresponding fields for the entry type "company" in the Names application is:

```
class "company" called "Company" of { "cardfile" "Names" } contains
    {sortOn := format "%s" { company }
    company: string called "Company"
    names: array called "Name" { entry called "Name" contains
            {honorific: string called "Honorific"
            first: string called "First Name"
            last: string called "Last Name"
            affiliation: string called "Title"}}
    addresses: array called "Address" firststoredin ll { entry called "Address" contains
            {address: string called "Street"
```

```
                    address2: string called "Street"

                    city: string called "City"

                    region: string called "Region"

                    postal_code: string called "Postal Code"

                    country: string called "Country"}}

        phones: array called "Phone" { string called "Phone" class "phoneClasses" with options {
        "titlefromclass" } }

        emailAddrs: array called "E-Mail" firststoredin email { string called "EMail" class "mailClasses"
        with options { "titlefromclass" } }

        notes: note called "Notes"

        cardType: integer called "Card Type"}
```

There are six fields for the class type company: Name, Address (with sub fields Street, City, Region, Postal Code, Country), Phone, E-Mail, Notes, and Card Type.

4. Cut and paste each field block to change the order of fields.

For example, if you wanted phone numbers to be listed before address(es), then you would change the class string and field order as follows:

```
class "company" called "Company" of { "cardfile" "Names" } contains
        {sortOn := format "%s" { company }
        company: string called "Company"
        names: array called "Name" { entry called "Name" contains
                    {honorific: string called "Honorific"
                    first: string called "First Name"
                    last: string called "Last Name"
                    affiliation: string called "Title"}}

        phones: array called "Phone" { string called "Phone" class "phoneClasses" with options {
        "titlefromclass" } }
        addresses: array called "Address" firststoredin ll { entry called "Address" contains
                    {address: string called "Street"
                    address2: string called "Street"

                    city: string called "City"

                    region: string called "Region"

                    postal_code: string called "Postal Code"

                    country: string called "Country"}}
```

5. Save the newton.adf file as a text file after making changes.