

Table of Contents

Getting Started with ParseRat.....	2
Delimited Input Lines.....	4
Fixed Format Input Lines.....	4
Page Image and Multi-Line Input.....	5
Web Form Files	8
Binary Files	8
Binary View	10
Parser Generated Fields	10
Name Parsing	10
Gender Evaluation.....	12
Street Address Parsing.....	12
City and State Parsing	12
Phone Number Parsing.....	12
Combining Input Fields	13
Splitting Input Fields.....	13
Date and Time Conversions.....	14
Julian Dates	14
Conversion from Multiple Input Fields (Parsing Corrections)	15
Using dBase Files.....	15
Mailing Options	16
The Output Tab.....	16
Selection of input records.....	17
Appending Results to an Existing File.....	18
Interleaving Output.....	18
Adding Header and Trailer blocks to the output file.....	18
Eliminating Duplicate Records	19
Saving Preferences.....	20
Command Line Interface	20
Hints and Tips.....	21
Examples in the Samples directory/folder	21

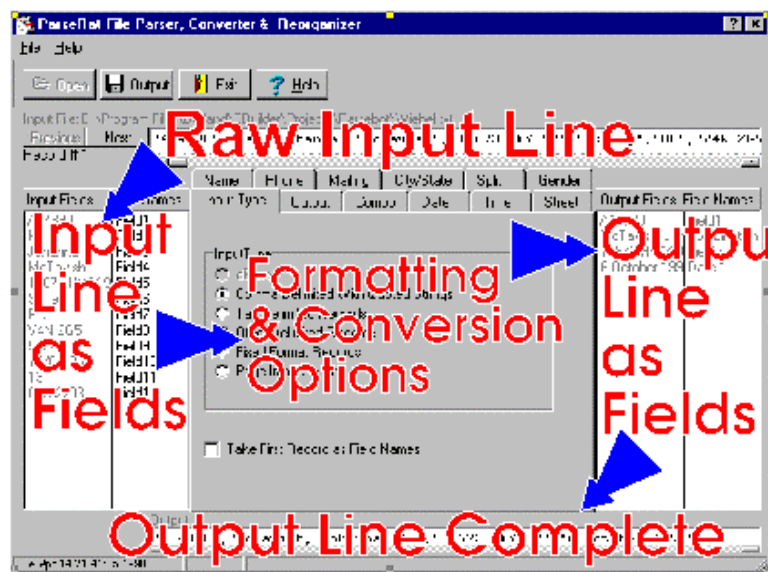
Getting Started with ParseRat

Description of system:

ParseRat works by:

- ❖ Reading an input file (or taking data from the system clipboard).
- ❖ Splitting up the input file's records, lines, data blocks or pages into fields based on user selection of input type (guided by ParseRat based on patterns it finds in the file).
- ❖ Processing those fields based on options set in the option tabs. This may include combining fields, splitting fields, converting fields in various ways or generating information about fields (e.g. the implied gender of a first name) into Parser Generated Fields.
- ❖ Creating output fields in the form and sequence specified by the user on the Output Tab, either by copying an input field directly or by using a Parser Generated Field.
- ❖ Changing the case (UPPER CASE, lower case, Proper Case), size (i.e. field width in characters), type (character, numeric, date) etc. of each output field if necessary.
- ❖ Creating an output record from the output fields.
- ❖ Writing an output file in a format specified by the user (or putting the data on the system clipboard).

The general processing flow is shown in the picture below:



Step by Step

- To start ParseRat, click on the Windows 95 Start button.
- Click on the ParseRat folder icon.
- Click on the ParseRat program icon.
- Click on the **Open** speed button or select **Open Input File** from the **File** menu (or click on the Paste From ClipBoard icon on the toolbar).
- Select the file to open in the dialog box which appears and click the **Open** button.
- ParseRat will display the first record in the area above the page tabs.
- It will choose a radio button on the **Input Configuration** tab and may select the **Comma Delimited** radio button and display the record as parsed out fields in the left hand columns if the input looks like "comma delimited text with quoted strings" (one of the most common requirements).
 - You may select a different radio button than the one chosen:
 - ❖ If the file is tab delimited click the **Tab Delimited** radio button.
 - ❖ If the file is delimited by a keyboard character (e.g. comma, space etc) click on the **Other Delimited** radio button and press the key for the delimiter to enter it into the box which will appear.
 - ❖ If the file consists of records in which the fields appear in constant positions on the line, click the **Fixed Format** radio button.
 - ❖ If you know that the input file is in Page Image form, click on the **Page Image** radio button (see below for more about page images). Note that if ParseRat finds FormFeed characters (ASCII 12) in the first 99 lines of the file, it will suggest Page Image input to you.
 - ❖ If the file is in binary format, click on the **Binary Files** button. ParseRat can extract the data from many binary files, even if you do not have their formal format specifications. Once you have selected the **Binary Files** option, all other input format options are disabled and to change your mind you must close and re-open the file. However you may view the first 15,000 characters of the file in binary form at any time by clicking on the **View Start of File as Binary** button at any time.
 - ❖ Note that the dBase radio button is controlled by ParseRat based on the extension of the input file. If the input file has a DBF extension, ParseRat will attempt to open the file as a dBase file and will parse its records accordingly.
- If any of the delimited formats is selected, the record will be immediately shown as parsed into fields in the left hand column.
- If you select either Page Image Input or Fixed Format Records you will need to indicate to ParseRat how to break up the lines of the input file into fields, following instructions in the appropriate topics. The Page tab is accessible only when the Page Image Input radio button is checked.
- Once the input record is parsed into fields, you may either go directly to the Output Tab or you may specify more specialized processing of the input fields by selecting one of the other option tabs.
- To see the effect of your settings on other records in the input file, click on the **Next** button to see more records.

The effect of making option changes is immediately shown to the user for the record being examined. You may use the "Next" and "Prev" buttons to examine the effect of your option selections on other records.

You may save the settings you make as Preferences using the "Save Preferences" function on the **File** menu item.

Input Record Parsed Into Fields. The left side of the screen contains the component parts of the input record after it has been broken up into fields and before further processing.

The rules for creating these fields from the input record are set by clicking radio buttons on the **Input** tab.

If **Page Image Input** is selected, this process is completed using the options on the **Page** tab which is visible only when page image input is selected.

The field names are either assigned by ParseRat or taken from the input file depending on the type of input file. They may not be changed on this screen by the user, but fields may be given user-selected names on the **Output** tab.

To select a field for further processing, first click on the destination - either an output field name or a source selection box on one of the filter/processing tabs **then** click on the input field name for the field.

Output Fields before Assembly. The right side of the screen displays the output fields resulting either from copying directly from input fields or from the inclusion of parser generated fields.

The field names are initially the same as the source names for the data, but may be changed by the user using the **Rename** option on the output tab.

It is the user's responsibility to ensure that the field names (if they are to be used in the output file) are acceptable to the applications in which they will be used.

Only for dBase output will ParseRat re-format output field names.

Options for formatting the individual output fields are found in the bounded box at the bottom of the output tab.

Output Record. The bottom of the screen area shows how the record will look after assembling the output records. The various delimiter options are selected from the bounded box at the top right of the **Output** tab.

To obtain help about any item on the screen, click on the title bar's ? (Question Mark) icon, then click on the screen item you wish to query.

Delimited Input Lines

Many files are received or generated in a "delimited" format. That means that the various fields are divided from each other by some form of separator or "delimiter".

Common delimiters are commas alone (,) for what are known as "**Comma Separated Values**" (.CSV).

If there is a possibility that the fields will contain commas, the "**Comma Delimited with Quoted String**" format is commonly used. In this format the fields are also separated by commas, but non-numeric fields are also surrounded by quotes ("). This is probably the most common delimited format. Care is needed to ensure that the "straight" quote is used, not the curly "typsetters'" quotes. The inclusion of additional quotes within the quoted strings should be avoided if you intend your output also to be in this format - although ParseRat will handle the situation correctly, many programs will be confused by it.

Values may be separated by tab (ASCII 9) characters for **Tab Delimited Files**.

Comma Delimited with Quoted String and **Tab Delimited** files are selected using specific radio buttons on the **Input** tab. Other single character delimiters (including **comma** and **space**) are selected by using the **Other Delimited** radio button and entering the delimiter character in the edit box which appears.

If an appropriate delimiter has been selected, the fields so defined will immediately be seen in the left hand column on the screen.

Fixed Format Input Lines

Instead of an input file having "delimiters" to break it up into fields, its elements may be in fixed locations on the line. ParseRat must be told what these divisions are.

Input Fields	Field Names	Next
//345 6789012 34567890123	Col1-5 Col6-12 Col13-End	//34567890123456789

To do this you use the mouse to set "tabs" (^) on the line below the input record. As you do this, you will see the parsed result appearing in the left hand columns as fields.

To delete a tab which you have already set, click on it again.

Fields generated by breaking up an input line in this way are given names describing their original positions. e.g. if two tabs are set after column 5 and column 12, the line will be parsed into three fields, with names like **Col1-5**, **Col6-12**, **Col13-End**.

Frequently you will not require all of these fields for your output record. That is no problem, just ignore the ones you don't want and omit them from your output specifications.

Should you redefine a field by altering tabs associated with it **after** you have used it as input to a parser generated field or as an output item that use will become invalid and the result will be blanked. Other fields not related to the changed tabs will not be affected.

The **Input Line Image** shows the input line being processed. For defining fields for Fixed Format input lines or for selected lines from Page Image input the user needs to set tabs below the line image. For full details see the items regarding **Fixed Format Input Lines** and **Page Image Input**.

Page Image and Multi-Line Input

ParseRat is able to parse out fields from page images produced by other programs. Frequently the only means of exporting data stored in proprietary formats is to use the capability of the original program to produce printed reports and to direct the report to a file instead of a printer. For example, a library system will always have the capability to print catalog cards, and that print file provides ParseRat with a source from which to export item data for use in some other application.

In other cases, you may need to extract data from multi-line blocks of data in a file.

In every case where input from multiple line blocks needs to be parsed, you will need to determine how the pages or data blocks are defined or delimited. Possibilities are:

- **Fixed number of lines:** Each data block has a given number of lines (typically this will be a file of address label information). *Select the **Page Image** radio button on the Input Type tab. Select the **Fixed Length** radio button and enter the number of lines per page or block.*
- **Form Feed:** Each page or data block (except possibly the first) begins with a FormFeed character (ASCII 12). *Select the **Page Image** radio button on the Input Type tab. Select the **FormFeed** radio button.*
- **Blank Lines between blocks:** Data blocks with no blank lines within a block, but with one or more blank lines between blocks. *Select the **Page Image** radio button on the Input Type tab. Select the **Blank Lines** radio button.*
- **Tag Text on Top Line:** The blocks are defined by some fixed text which occurs on a line at the head of the block. This text must be used ONLY in this location and is case-sensitive. This is often the situation with Emails in a file, where there is some constant text pattern in the header of each message. *Select the **Page Image** radio button on the Input Type tab. Select the **Tag Text on Top Line** radio button and enter the tag target text. Note that if there are lines ahead of the first line with this text string, they will be ignored.*

You have the option of specifying that the tag text may appear anywhere on the line, or only at the left hand end. This latter option is particularly useful for print image files which were originally intended for a line printer of the 1403 pattern, where the first character on the line is for vertical carriage control ("VFC" files). To set up to find the top of such pages, enter a figure 1 in the tag text box and select the "At left only" option.

The location of fields in the page or block can be either fixed or tag-related. If they are fixed, they must always appear in the same position relative to the top line of the page as defined above. If they are not fixed, refer to **Tagged Page Input**.

The example below is of one of the more complex forms, which has headers, footers and several repeating data blocks both down and across the page.

Each data block is intended to result in one record with each such record able to contain data from the header and footer area as well as from the data block.

Most page input situations will be simpler and will frequently comprise files where there is no header or footer, just data blocks, one across with no gutter (e.g. a file intended to be printed on continuous form mailing labels) or a file with headers and footers, but with single row data blocks each containing several fixed columns of data (e.g. a report print file from another program).

Street Name Phone Mailing City/State Split Gender

Input Type Page Output Combo Date Time

Header Line 1 Page 1
Header Line 2 Sheet 1
Header Line 3

01234567890	abcde fgh i j k	ABCDEF GHI J K
abcde fgh i j k	ABCDEF GHI J K	01234567890
ABCDEF GHI J K	01234567890	abcde fgh i j k
lmnopq rstuv	LMNOPQ RSTUV	01234567890
01234567890	lmnopq rstuv	LMNOPQ RSTUV
LMNOPQ RSTUV	01234567890	lmnopq rstuv
01234567890	abcde fgh i j k	ABCDEF GHI J K
abcde fgh i j k	ABCDEF GHI J K	01234567890
ABCDEF GHI J K	01234567890	abcde fgh i j k

Footer Line 1 of Page 1
Footer Line 2 of Page 1
Footer Line 3 of Page 1

Selected Line 10 of 20
on Page 1
Horiz. Band 2 Row 2 Vert. Band 3

Set heights of page bands:

Header Above Line 5
Data 4 Row Bands
Footer Below Line 16

Vertical Data Bands

Left Gutter 5 Width of Data Blocks 13
Number of vertical bands 3

When you click on (or select with the arrow keys) a header or footer line, that line will be displayed in the input record area at the top of the screen. You define the fields by setting tabs in the same way as for fixed format input lines. When you click on line in a page data block, that line from that block will be displayed in the input record area for field definition.

Remember that each data block will result in an output record containing fields derived from that data block, together with fields from the header or footer area if you have defined and selected any.

If you select the **Tagged** page format, you may enter such tags in a list box. ParseRat will then search the page for the **first** (or other specified) occurrence of text which **exactly matches** the tag and will enter the text following it into an input field which is given as a name the tag itself. The data field will contain all data on the line following the tag **up to any other defined tag found on the line**.

You may combine the selection of fixed location data and tagged data in the same run, but you cannot handle multiple data blocks on a page at the same time as tagged data.

If ParseRat recognizes the incoming file as XML, it will make an initial guess at the tags you need.

- Open the input file as described in Getting Started with ParseRat.
- If the file is of fixed length pages (i.e. there are no FormFeed - ASCII 12 characters defining the page ends) enter the page length in lines in the box which appears when you select **Page Image Input** and **Fixed Length**. If there are FormFeeds (ParseRat will tell you if it found any in the first 99 lines) and you wish to use them to determine page ends, select **FormFeed ASCII 12**. If there is some unique character string which defines the start of a page (and for this option the page need not correspond with a physical page - there could be several "pages" on one physical page), enter that character string in the box which appears when you select **Tag Text on top line** (note that the string must match exactly including any punctuation, spaces and capitalization). Any lines ahead of the **first** occurrence of the string will be ignored as ParseRat establishes the top of the first "page" where it finds the first top line tag.
- Select whether you are processing fixed format or tagged pages. Note that you can process some fixed format data as well as tagged data using the **Tagged** option.

- Click on the **Page** tab to show the page image of the first page. Note that the characters are sized to fit the form on the tab, so if it is a very wide page they may be too small to read but only big enough to see the page layout. The top line will be displayed in the **Input Record** area near the top of the screen.
- If a fixed format page includes headers (including blank lines) above the first line of data records, place the cursor in the top line of data records and click the Header button. The header area will be displayed in yellow.
- If a fixed format page includes footers (including blank lines) below the last line of data records, place the cursor in the bottom data record line and click the footer button. The footer area will be displayed in green.
- If there is more than one data record (or "Data Block") per page, place the cursor in the bottom row of the top data band (for many reports being imported with one row per item, this will be the same as the top data row). Then click on the data button. The data block will be displayed in blue (but for single-row records you will not see it). For the tagged data option, the whole "page" is considered a single data block.
- If there is more than one record per line across a fixed format page, complete the Vertical Data Band box at the bottom right of the page.
- Place the cursor in the first line from which you wish to extract data (it might be a header area, a footer area or a data block) and notice that the line appears in the input record area.
- Set tabs (^) using the mouse to define the input fields. **If you want the whole line to be a field, set a tab at or beyond the right-most position on the line.** Note that ParseRat displays the column count of the cursor position.
- Repeat for every line from which you wish to extract data.
- Once the input record is parsed into fields, you may either go directly to the Output Tab or you may specify more specialized processing of the input fields by selecting one of the other option tabs.

A **Page Header Area** is that area at the top of a page which either contains no information to go into the output record, or contains information common to all records generated from that page. **It is displayed in yellow.**

To define a header area, click on the top line of non-header data (the line **below** the last header line) and then click on the **Header** button. If there is no requirement for a header area, click on the top line of the page and then click on the **Header** button.

A **Page Footer Area** is that area at the bottom of a page which either contains no information to go into the output record, or contains information common to all records generated from that page. **It is displayed in green.**

To define a footer area, click on the bottom line of non-footer data (the line **above** the first footer line) and then click on the **Footer** button. If there is no requirement for a footer area, click on the bottom line of the page and then click on the **Footer** button.

Remember to check your data band height (in rows) after setting a header or footer area.

To define data blocks, first define any header or footer areas. Then click on the **bottom** row of the top data block and then click on the **Data** button (note that for single row blocks, the top and bottom row are the same).

When you click on any area in a page data block, **the whole block is displayed in blue** but note that if the data block is only one row high, you will not see the blue background because it will be covered by the selection bar.

A **Page Data Block** is an area on the page which will result in a single output record.

It may consist of one or more rows of data. There may be more than one data block on the page, both horizontally and vertically.

To define data blocks, first define any header or footer areas. Then click on the **bottom** row of the top data block and then click on the **Data** button (note that for single row blocks, the top and bottom row are the same). If there is more than one vertical band of data blocks across the page (e.g. a label file with more than one-up) they should be defined using the area at

the bottom right of the page. The (blocks must be of identical format). The “gutter” if any is the number of blank columns at the left of the leftmost block.

When you click on any area in a page data block, **the whole block is displayed in blue** but note that if the data block is only one row high, you will not see the blue background because it will be covered by the selection bar.

Web Form Files

Forms filled out on web pages result in coded files being dispatched to the server or to an email address. These files have a specially code format. ParseRat is able decode the format and present the fields as if they were tagged fields for your further processing.

Since web form "checkbox" tags are inserted in the file only if the box is checked on the form you need to be sure that ParseRat "knows" about all of the check boxes. ParseRat will learn about the web form as it moves through webform records in a file, but it will only be able to output fields based on tags that are present when it starts the output because only these will appear as input fields.

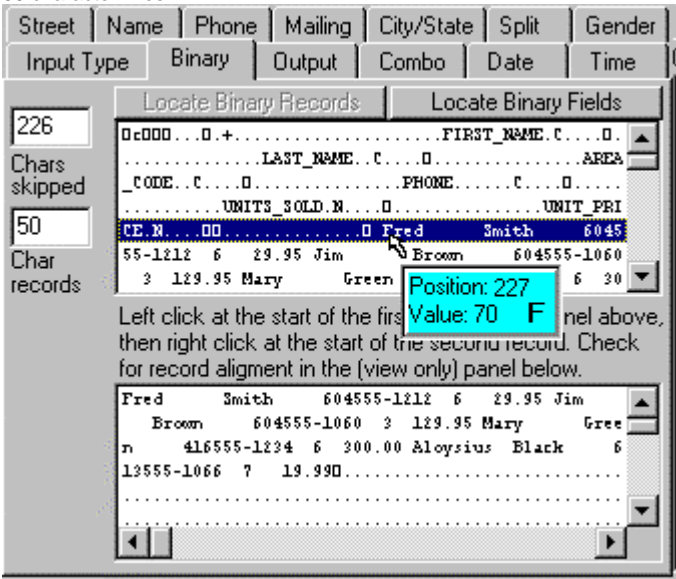
The easiest way to capture data from all checkboxes is to check all boxes on the form, then send the file and open it in ParseRat. ParseRat will recognize all of the tags.

Then Save Preferences with those tags showing in the input area. Now, when you load those saved preferences, ParseRat will be able to recognize all the tags.

If you wish to extract data from downloaded copies of HTML web pages, you may do this as tagged input.

Binary Files

Many files consist of binary data, with no delimiters. Frequently such files consist of an initial header block, usually giving details of the structure of the remainder of the file. Following the header (if any) are records of a fixed length but with no delimiters. These records may consist of text-only material or binary numbers or a combination of both. Even if you do not know the formal structure of the file format, in many cases you can determine the structure of a given file by inspecting it. You may then export its data in a manner similar to that for fixed format files. Selecting the "Binary" radio button opens a tab permitting you to choose the number of header characters to skip and the size of the records following the header. You may enter those numbers into the appropriate boxes if you know them but more probably you will find them by experimenting with the mouse. When the binary tab opens, it displays two windows. The upper window shows the first 15,000 bytes of the file wrapped into 50 character lines:



Moving the mouse cursor over that window displays the character position and the value of the character at that position. Clicking the **left** button of the mouse on one of these characters will set that character as the first character of the first record - those characters ahead of it will be treated as a header block. The remainder of the file block is shown in the lower window as

records of the size set in the appropriate box. The alignment of columns in this view provides a check on your record size selection.

Clicking the **right** button of the mouse on a character further into the file will set that position as the beginning of the second record - this defines the record size. In the example shown, right-clicking on the **J** of Jim changes the record size to 43 characters, and displays the structure in the lower window, as shown:

Fred	Smith	604555-1212	6	29.95
Jim	Broom	604555-1060	3	129.95
Mary	Green	416555-1234	6	200.00
Aloysius	Black	613555-1066	7	19.990

The alignment of the fields shows that the correct record size has been selected. The records now appear in the input record

Previous Next Fred Smith 604555-1212
Record # 1

window [set tabs below line image](#) Street Name Phone Mailing and may be treated in the same manner as fixed format records, defining fields by setting tabs below them.

If the fields contain only ASCII text, they may be treated exactly the same as fixed format records. However, a binary file frequently contains binary numbers, like the field in columns 17-20 of the example below (note that Intel-pattern binary numbers have the high-order bytes at the right hand end). A valid length for a binary number is 1, 2, 4 or 8 and the appropriate type may be chosen by clicking the **Locate binary fields** button on the **Binary** tab. ParseRat displays the value of the field in each of the valid configurations against the corresponding radio buttons as shown:

3 Input Fields Field Names
FIRST_NAME Col1 16
10 Col17_20
Col21_End

Input Type Binary Output Combo Date

Locate Binary Records Locate Bin

Binary Field Definition
If the selected input field is 1,2,4 or 8 characters long, it could have a binary numeric value as shown. To use that value in processing this file, select the format you wish to use.

Input Field Name: Col17_20
width: 4

☐ ASCII text string
☐ Signed Integer 10
☒ Unsigned Integer 10
☐ Floating Point 0.000000

Note that binary numbers may be stored with the first byte being the highest value byte, or with the first byte being the lowest value byte and you may select which conversion you wish by using the radio buttons:

1st Byte
☒ Lo
☐ Hi

The default shown is known as "Little Endian", a computer architecture in which, within a given 16- or 32-bit word, bytes at lower addresses have lower significance (the word is stored "little-end-first"). The PDP-11 and VAX families of computers and Intel microprocessors and a lot of communications and networking hardware are little-endian.

Big Endian is a computer architecture in which, within a given multi-byte numeric representation, the most significant byte has the lowest address (the word is stored "big-end-first"). Most processors, including the IBM 370 family, the PDP-10, the Motorola microprocessor families, and most of the various RISC designs, are big-endian.

Zoned Decimal is a holdover from punched cards, where the sign of a number was indicated by punching in one of the top two rows of the card above the low order digit to indicate plus or minus. This translates into numbers which look normal except for the last digit. If there is no overpunch, the number looks normal. (Unfortunately, the zoned decimal format does not translate very well to the ASCII code, since ASCII uses different codes to represent the digits and letters. What happens is that on most ASCII machines, the same characters are used, even though the hex codes for them are different and ParseRat follows that convention.)

e.g.:
000001160} would convert to -116
000003407N would convert to -340.75
000003407S would convert to 340.75
0011693450 would convert to -116934.56
0000000395 would convert to 3.95

The decimals result from an assumed decimal point common in COBOL programs dealing with currency. In ParseRat you would parse it out into cents, then use the /100 option of the Output Tab to convert to dollars.

Another possible "binary" option is "**packed decimal**", where each byte except the last carries **two** numeric digits and the last byte carries one digit and optionally an indication if the value is negative. This will frequently be found when converting data created by COBOL programs.

You may have noticed that the example given is of a file in the dBase format. ParseRat will handle dBase DBF files as a defined input type, but other database files which have not been pre-defined may be handled by ParseRat in this manner. For you to practice with, the example file is included in the Parserat\Samples directory as SALES.DBX and the dBase file from which it is copied as SALES.DBF.

If you need to extract more than 99 fields from a structured binary record, you will need to do this with multiple passes.

EBCDIC, or Extended Binary Coded Decimal Interchange Code is the format in which non-numeric data is stored on some systems (principally IBM Mainframes). Many programs which transfer data between EBCDIC mainframes and ASCII systems translate each byte as if it were a single text character. This means that binary numbers and packed decimal numbers are wrongly converted. If the EBCDIC Source box is checked, ParseRat will correct for this when handling number fields on the Binary Tab.

Binary View

Clicking on the "View Start of File as Binary" button displays a panel on the **Input** tab which gives a binary view of the first 15,000 bytes of the file. Clicking on the "Display Input Format Options" button returns you to the input option selection. In the binary view, undisplayable characters are shown as small squares. To see their value, move the mouse pointer over them. Common values are:

0: NULL - you will usually see this only as part of a binary number.

9: Tab

10: LineFeed

12: FormFeed

13: Carriage Return.

Lots of little squares with values of over 127 usually indicate a file with lots of binary numbers, which you will probably want to process as a binary file.

Parser Generated Fields

In addition to assembling selected input fields into different sequences and formats for output, ParseRat can process input fields and generate fields which result from the processing.

For example, it can take in a field or group of fields representing the "Street" line of an address and create individual fields for Suite Designator, Suite Number, Street Number, Street Name, Street Designator (Ave, Rd, St, etc), Street Direction (e.g. N, S, SE), Odd/Even property of street number and a field for "Non-Civic" address data (RR, PO Box etc).

As another example, it can take in a date in a variety of formats and generate not only an output date in a different format, but also the day of the week it represents and integers for Year, Month and Day.

When processing Page Image data using the **Tagged** format, ParseRat will automatically extract the first four URLs on the page (defined as something with :// in it) and the first four **Email addresses** (defined as something longer than six characters and including an @ sign).

A Parser Generated Field is placed in an output field using the Output Tab by first adding or inserting an output field, or by clicking on the name of an existing output field (to change the definition of that field). The name of a parser generated field is then clicked in the Parser Generated Field list box.

Name Parsing

ParseRat has the ability to separate people's names into their components (Title, First Name, Middle Names, Last Name and Suffix), based on sequence data specified by the user.

This example shows how you may set ParseRat to extract name data suitable for use in mailings from a telephone listing of the form commonly available on CD Rom.

The left hand block of "Radio Buttons" is set to tell ParseRat that the name elements (if present) in this example will be in the sequence LastName, FirstName, MiddleNames, Suffix, Title. The Suffix and Title entries must be found in the lists specified (which you may edit on-the-fly).

ParseRat knows about multi-word last names, both hyphenated and non hyphenated. One and two word prefixes which may be used with surnames are defined on the **Mailing** tab. By this means a last name such as "Van De Geest" will be correctly parsed into the Last Name field.

In addition to providing the separate name elements for output as individual fields, ParseRat will also create a "Standardized" name field, placing the elements in the sequence you specify using the right hand block of "Radio Buttons".

This example also shows another feature of ParseRat, in that "noise" words may be ignored in parsing names (to avoid things like "Children's Phone" or "Fax Line" being included as part of a name). These noise entries are specified on the **Mailing** tab as items to strip from incoming names - they too may be edited by the user.

If a title is not found in the incoming name field(s), ParseRat can generate a default male or a default female title based on the implied gender of the first or middle names (see the **Gender** tab).

To assist in customized mailings, ParseRat produces a parser-generated field called **Salutation** based on the following rules:

- If there is anything in the title field, that is used as the Salutation.
- If there is nothing in the title field the first and/or middle names are used as the Salutation, separated by "&" if they were so separated in the raw input.
- If there is nothing in either title, first or middle name fields, the default male and default female titles are used separated by " or ".

As with certain other tabs, ParseRat can assemble a field to parse from several incoming fields since initial entries may have been incorrectly parsed, or not parsed into enough components.

Business Names should be handled separately from people's names (if exporting from a CD-ROM, export the business listings and residential listings separately). Usually, you will want to add "The" to the title list since it frequently comes at the end of the listing. You will usually set the radio buttons to parse for only Last Name & Title. This will have the effect of only altering the listed value by moving items in the "title" list to the beginning of the "Standard Name" field.

Name Input Sequence ("Sequence In") buttons are used to tell ParseRat the sequence in which name elements will appear, if present. The button in the left hand column of buttons level with the element which will be first in the input should be checked. The button in the second from left column level with the item which will appear second should then be checked, etc. If it is known that an element will not be present, no button level with it should be checked. The radio buttons for any unused button columns should be "parked" at the top of the column. If the Title or Suffix elements are used, elements will appear in the appropriate parsed fields **only** if they match entries in the Title or Suffix lists provided (which you may edit on-the-fly).

Name Output Sequence ("Position Out") buttons are used to tell ParseRat which elements to use in generating a Standard Name as a Parser Generated Field, in addition to the individual parsed out elements.

Infer Title From Gender instructs ParseRat to look at the elements parsed out as First Name and Middle Names and to assign an appropriate male or female title if the title field is blank after parsing the name. The male and female titles are set on the **Gender** tab and changes may be made to the pre-programmed genders defined for particular names on that tab.

Suffixes to names are items which normally appear after the last name (like Sr, Jr, Esq, BSc, PhD, III, etc) which appear in this list. To edit an item, select it by clicking on it and then you may delete it, or change it. Note that the matching for parsing purposes is done using the left hand column in a case-insensitive manner (i.e. ignoring capitalization), but the item in the right hand column is what goes into the parser generated field. This is to enable the user to standardize terminology, the case of the letters, etc if desired.

Titles to names are items which normally appear before the first name (like Mr Mrs, Miss, Ms, Rev, Prof, etc) which appear in this list. To edit an item, select it by clicking on it and then you may delete it, or change it. Note that the matching for parsing

purposes is done using the left hand column in a case-insensitive manner (i.e. ignoring capitalization), but the item in the right hand column is what goes into the parser generated field. This is to enable the user to standardize terminology, the case of the letters, etc if desired.

Gender Evaluation

ParseRat is able make judgements about gender based on the first names it is given. It is able to genderize up to three incoming first name fields directly, as well as based on first name or middle name fields generated using the **Names** tab. Additionally, it is able to set Gender 1 based on the title field parsed out under the names tab, since a name such as Mrs. V. T. Singh might appear in the input, without a first name to genderize.

The Gender tab allows the specification of one default male and one default female title.

Additionally, there is provision for a "joint" title - mainly for use in mailing list situations where the presence of names of more than one gender usually indicates a couple. The user may, of course, change these default titles.

For performance reasons ParseRat has 5,000 names directly coded into its programming, but you may add names or override the definitions of existing programmed names. Be aware that many names (e.g. "Robin" do not reliably indicate gender).

To check on how a name has been already defined, enter it into the appropriate edit box and then press the **tab** key.

Street Address Parsing

ParseRat will parse out the components of a street address into Parser Generated Fields.

It will also create a "Standardized" street address in the sequence which you specify using the block of radio buttons on the right hand side of the tab.

For example, you may not wish to have non-civic address information in the "Standard" line, but export it to a different field. In that case, ensure that none of the radio buttons level with the non-civic field is checked.

Some Street and Suite designators (e.g. Street, Avenue, Road; Suite, Ste, Unit, Apt) are pre-programmed but may be overridden or supplemented by using the lists on the **Mailing** tab.

Since some postal authorities have preferred forms or abbreviations, ParseRat provides the means to substitute these preferred forms for suite and street designators.

City and State Parsing

Frequently, address data (e.g. when using page image input for a file intended as a label print file) the data for City, State/Province and Postal/Zip code will appear in one field. ParseRat provides the means to parse these out into City, State/Province and Postal/Zip Parser Generated Fields.

The alpha-numeric pattern for many Postal/Zip codes is recognized.

For North American addresses the parser generated field for State/Province will contain the approved abbreviation if the fully spelled out version is found. The user may edit the State/Province information using the list box provided.

For addresses having a "British" postal code pattern, the abbreviation "UK" will be entered in the "State" field.

An attempt is made to recognize "European" postal codes by structure and context.

Phone Number Parsing

Telephone numbers may be presented in one or two incoming fields. They will be parsed into two fields designated as "Area Code" and "Phone" parser generated fields.

For some non-North American phone numbers the generic "Splits" and "Combination" tabs allow parsing of phone numbers into components.

Combining Input Fields

Input Fields	Field Names	Combination Component Specifications
A14890	Field1	C1= <input type="text" value="Field4"/> <input type="text" value="Field3"/> <input type="text" value="Field2"/>
Ms	Field2	
Johanna	Field3	
McTavish	Field4	

Combination Results	
Combination 1	McTavish, Johanna (Ms)

ParseRat is able to combine a set of input fields into a single output field, as in the above example. By default the incoming fields are separated by single spaces, but this may be overridden by specifying up to two characters for each delimiter (a valid value is no delimiter).

To specify a combination, click on a white field in the Combination Specification area, then on the appropriate input field name. Repeat this for each of the white fields in the Combination Specification area.

To specify a delimiter (separator) to use between two fields, click on the yellow field between them, then enter the desired separator in the yellow edit box and press the tab key. Repeat this for each separator.

Splitting Input Fields

Input Fields	Field Names	Source 1: AUTH_C	Source 2: CIS_PP	Source 3: OTH_EMAIL
49-6126-93100	AUTH_CON			
100045,3651	CIS_PPN			
mbohnnet@mb	OTH_EMAIL			

Split	Field
Split 1A	49
Split 1B	6126
Split 1C	93100
Split 2A	100045
Split 2B	3651
Split 2C	
Split 3A	mbohnnet
Split 3B	mbmarktcons
Split 3C	com

Sometimes it is necessary to further split an input field. ParseRat will perform such a split using the **Split** tab. Up to three such fields will be split into up to three fields, based on alphanumeric "words" separated by spaces or other delimiter characters.

The delimiter options are commas(,), slashes(/) or the default which splits on **any** non-alphanumeric characters.

The resulting splits contain only alphanumeric characters if using the default setting (note that "splitting" into one field has the effect of creating a new field by simply stripping out all non-alphanumeric characters - which may be desirable in some circumstances).

(The above example shows a section of a dBase file with a German telephone number, a CompuServe ID and an Email ID. Whether these are likely candidates for such a split is dependent on the user application.)

To specify a split, click first on the Source selection box, then on an input field name.

If it is desired to further split one of the fields resulting, you might consider two passes of the program, creating an intermediate file.

Date and Time Conversions

Source	Date	Weekday
(Source 1)	Date 1	Weekday 1
(Source 2)	Date 2	Weekday 2
(Source 3)	Date 3	Weekday 3
(Source 4)	Date 4	Weekday 4

ParseRat is able to convert dates from many different formats into several standard formats. It can also handle dBase date fields both as input and output.

For input it is necessary for the user to specify **only the sequence of the incoming date elements** (for non-dBase date fields) and whether the incoming field also contains time data. You do not need to tell it about dBase date fields because they are always in a pre-defined format. ParseRat can recognize dates in numeric format with or without delimiters and in alphabetic format in English, French, Spanish, German and Italian. If any non-DMY data (e.g. the day of the week) is prefixed before the date, ParseRat will ignore it since it calculates day-of-week from the day/month/year values. This means that non-English day-of-week data will not cause confusion. Should you wish to extract and use such date prefix information, use the SPLIT filter tab on the field in addition to the DATE filter tab.

If the date field might have time information also included, check the box so marked. Time information will be parsed as for times defined under the "Time" tab.

Year 2000 note: If the incoming year has only two digits, the user may specify the year prior to which dates will be considered to come after the year 2000 in generating 4 digit output years. e.g. if you specify 15, years from 15 to 99 will be assumed to refer to 1915 to 1999 and 00 to 14 will be assumed to refer to 2000 to 2014.

The user selects the format of output dates, with the current date showing in the chosen format on the page as visual feedback. ParseRat also generates integers for year, month and day which may also be output as Parser Generated Fields.

ParseRat handles time conversions either when the time is part of an incoming date field (the user specifies this option by checking the appropriate box on the Date tab) or when there are separate incoming time fields.

As options, ParseRat can treat incoming time values as a decimal fraction of a day (often time values are expressed as Julian dates plus a decimal portion) or as the number of seconds since January 1, 1970, commonly used in computer systems and known to C programmers as "Calendar Time". In the case of Calendar Time, ParseRat will also complete date information in the appropriate fields, unless another specific date source has been selected.

Times will be correctly handled whether or not they contain delimiters and will correctly handle am/pm designations. Times are parsed into integer parser generated fields for hours (24 hour clock), minutes and seconds.

A time which comes in as part of Date 1 or Time 1 will be parsed into hour 1, minute 1 and second 1.

If ParseRat finds a seven digit number in a date field, it treats it as a Julian Date, since no other date format fits that pattern.

Should more date formats than dBase plus one other input sequence and one other output format be required for a single file, this may be achieved by using more than one pass of the program, creating an intermediate file.

Julian Dates

Astronomers and chronologists use a system of numbering days called Julian days or Julian day numbers.

The Julian day system was developed by Joseph Justus Scaliger (1540-1609) in the course of his work on ancient chronologies published in 1583, and probably named for its fit to the Julian year.

January 1, 4713 B.C. is the date determined by Scaliger to have three different yearly counting schemes simultaneously at the start of their sequences and the Julian Date of a day is the number of days since that date. This makes it easy to determine the number of days between two days (just subtract one Julian day number from the other).

The system of Julian days should not be confused with the simpler system of the same name which associates a date with the number of days elapsed since January 1st of the same year.

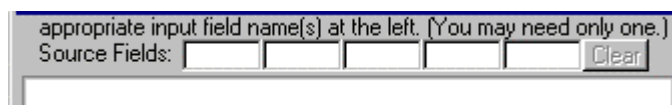
The Julian Date can be stored in a 32 bit integer (taking up only four bytes), less space than needed for most other date storage (including the YYMMDD six digit format which brought the "Millenium Bug" to the world).

Conversion from Multiple Input Fields (Parsing Corrections)

ParseRat accepts instructions to put a specific item in a particular place by the user clicking first on the **destination** and then on the **source**. When the user clicks on an already assigned destination, the current source is highlighted ("Selected") but the user may click on a different source to change that selection.

On several tabs, the user may assemble a composite input field to parse by selecting several input fields which will be strung together.

This is particularly the case with name and address data, where the elements of a name or a street address may be already partially parsed into several input fields. (This parsing may not have been performed correctly originally, or further processing may be needed).



Click first on the leftmost of the boxes marked: Source Fields, then click in the Field Name column on the field name of the first input field which you wish to assemble for parsing (frequently you will need only one such field).

Then click on the next of the boxes marked Source Fields, then click in the Field Name column on the field name of the next field to be assembled for this parse, etc., etc.

The composite field being assembled will be displayed below the Source Field boxes.

To clear an entry from the Source Field boxes, click on it and then click on the Clear button.

Using dBase Files

The dBase file format is read and written by most database and spreadsheet programs. dBase files contain field names and the fields are already tagged as numeric, character, date etc. ParseRat will not recognize dBase "Memo" fields or binary or picture objects, but will handle the other fields in a table which contains them.

On output ParseRat will write dBase type III compatible files if that option is selected. The user has the choice of writing dates as dBase date fields or as character fields.

If an output field is set as "Logic" format, only values of "?", "Y", "N", "T" or "F" will be permitted in that field and other values will be converted to T or F based on the following rules, applied sequentially:

- 1 If the first alphanumeric data in the field is a number not equal to zero, the field will be set to "T".
- 2 If the first alphanumeric data in the field is a number zero, the field will be set to "F".
- 3 If the first alphanumeric character in the field is "Y", "y", "t" or "T", the field will be set to "T".
- 4 If the first alphanumeric character in the field is "N", "n", "ft" or "F", the field will be set to "F".
- 5 If the field has not been set to "T" or F, it will be set to "?".

Note that if you are generating a new dBase logic field, you may pre-set it to "?", "T" or "F" by setting it to a parser-generated Empty Field, and then setting a default value of "?", "T" or "F".

Mailing Options

The **Mailing** tab allows the user to edit various options used in processing mailing lists.

- ❖ Street Designators and Approved Abbreviations
- ❖ Suite Designators and Approved Abbreviations
- ❖ Non Civic address elements and Approved Abbreviations
- ❖ Items to trim from "Telephone Directory" names (e.g. "Fax Line", "Teen Phone", etc)
- ❖ Capitalization Exceptions, for use in "Proper Casing" words
- ❖ Surname prefixes to allow the correct parsing of multi-word non-hyphenated names (e.g. "De La Rue", "Di Angelo", "Van Gelder").

The Output Tab

This tab controls how data is placed into the output file. When you first open the output tab, a blank output field is created and you are asked to select the item to go into it. You do this by clicking on either the field name of one of the input fields or on one of the items in the "Parser Generated Fields" area. The field name will change from generic "Output XX" to the name of the item selected.

Output fields are limited to 1255 characters and output field names to 31 characters.

You may (and probably will) wish to change the field names if your output is to be used in a situation in which fields names are significant. **It is the user's responsibility to ensure that field names are valid for use by other programs which will use them.**

*In many cases you will wish to copy all of the field names and contents from the input fields to the output fields, and then make minor changes. To assist in this process, a button is provided on the **Input Tab** which performs this if appropriate. This button is visible and enabled only when you have defined input fields, but not output fields.*

To add another field, click on the Add Field button if you wish to place a new field at the end. If you wish to place it above the currently selected field, use the Insert Field button. The Delete button will delete the currently selected output field. You may set many options for the currently selected field by using the functions in the framed **Detail Setup For Field** area at the bottom of the page. If you are intending to save the file as a dBase file, or in a delimited format which includes field names you may wish to rename the field.

When you click on the dBase output format option, the field names will be converted to valid dBase field names. ParseRat will also check that you have specified field widths and decimal places as required for dBase files. If the selected field is a date field, you will be offered the option to save it either as a dBase "Date" field or to save the format you see in the list.

You will be offered case conversion options for character fields and the option to convert to Soundex coding instead.

(**SOUNDEX** coding was developed around the year 1900 for use with spelling variations in similar sounding names in such things as the US Federal Population Census. It is still useful in processing names which sound similar (particularly when searching for duplicates in a large database of names). The algorithm will convert words into a pattern of one letter and three digits, which represents the "sound" of the word. McDonald and Macdonald have the same Soundex code as do Danyliu and Donnelly. If you wish to do a duplicate elimination and you are not sure that all names are spelled correctly, you may include the soundex values of the name field in addition to its regular representation.)

If you specify that a field is to be numeric you will be allowed to specify decimal, hexadecimal or scientific output format.

Additionally you may elect to have the number divided by 100 by checking the "/100" box, to permit the conversion of percentages to ratios, cents to dollars and to handle any field with an "assumed" decimal point two places from the right (common with COBOL currency data of the form 9(5)v99). This can also convert street addresses to block numbers for mailing and canvasser operations.

If a numeric field is of a fixed width, the user has the option of having it filled out with leading zeros. For "Comma Delimited with Quoted Strings" output, numeric fields are not surrounded by quotes.

The Output Tab also provides the means to eliminate duplicate records, based on specified matching fields in the output record.

You may re-sequence the output fields by dragging the field name items to new positions by placing the cursor on the field name of item to be moved and by dragging it to a new position by holding down the left mouse button while moving the cursor to a new position. When you release the button, the move takes place.

Should you wish to insert material before or after the records created from the input material you may specify Header and Trailer Blocks to go into the output file.

Only after you have viewed the Output Tab to specify output formats will you be able to Save the Converted File, that option is disabled until you have done this. You may start the output process either from the File menu or by clicking the Output icon on the toolbar (the one with the picture of a disk on it). If you wish to test a sample of your file, you may check the "1,000 record test" box. If that box is checked, the program will terminate after writing 1,000 output records which you may then test with your other applications before committing to a run which may produce millions of output records. The evaluation copy of the program is limited to this 1,000 record test.

Instead of naming an output file, you may select "Clipboard" as an output format option. In that case the output will be copied to the clipboard.

Proper Case Conversion is the conversion of character fields to a format in which initial letters are capitalized and other letters are not. This will not always result in the correct capitalization so exceptions have been specified. For example a word beginning with "Mc" usually requires the next letter to also be a capital as in "McDonald". Some words (usually trade names) like "WordPerfect" also require to be capitalized using special rules. ParseRat provides the means to specify up to 50 "Capitalization Exceptions" on the **Mailing** tab. Any word beginning with (or comprising) a letter sequence included in this list will have the capitalization pattern established according to the list. This also permits the initial letter to be lower case if so specified in the pattern. If a letter sequence is followed by the < character its capitalization pattern will be used only when standing alone as a word. e.g. the pattern "IV<" when applied to KING GEORGE IV would result in King George IV, but when applied to AUNT IVY would result in Aunt Ivy and not Aunt IVy. The list may be edited "on the fly" by the user if desired.

A **Default Value** may be specified for a field. If the value in the field is blank, the default value (if one is specified) will be used instead. To specify a default value for a field, click on the "Default" button and enter the value in the adjacent box. This mechanism can be used to set a field to a constant value by clicking on the field name in the column at the right and selecting "Empty Field" from the list of Parser Generated Fields. In such a case the default value will always be the one used. This can be used much like a simple "Mail Merge" to create repetitive lines incorporating items from the input records (some of the C++ code used in programming the ParseRat application was created that way).

The **Duplicate if Blank** setting is used to "fill down" if fields are blank. If it is set and the field is blank, the value of that field from the previous record will be used. If a default value is also set for the field, it will be effective only until a non-default value appears in that field.

The **Rename Field** option allows the user to change the name of the selected field. This is usually only of value if the output file is to contain the field names in some manner.

The **Remove Leading/Trailing Blanks** option is set for all fields by default but may be turned off by the user for the selected field. Its effect is to strip both leading and trailing blanks from information in the incoming fields before they are further processed. Trailing blanks will be added if the field width value is set to more than the width of incoming data for that field.

The **Field Width** setting sets the width of the selected output field. If longer than the setting it is truncated. If shorter it is padded with blanks.

File Output Formats may be selected by clicking on the appropriate radio buttons.

Note that if dBase is selected you will be required to ensure that field widths and decimal places have been defined where appropriate.

The WordPerfect setting will create a WordPerfect version 4.2 Secondary Merge File, which may be used by that and later versions of WordPerfect.

Many software packages, including **Microsoft Word** will perform a mail merge using a file which is **Comma Delimited with Quoted Strings**.

For Microsoft Word merge purposes you should also check the box to output the field names as the first record, since this will allow more flexibility within Word and will allow you to create a master document which will work with different data files, since it uses the first record to specify merge fields to use.

(Some software packages will use the first record of a delimited file to obtain or store field names. For example, Word for Windows can use a file written in this format as a data file for mail merge purposes. ParseRat can deal with a delimited file of this form both for input and output.)

Skip if Empty. If the "Skip if Empty" attribute is set for an output field, ParseRat will not output the record containing that field if the field is totally empty. Note that if the input field on which it is based contains only spaces, they will normally be eliminated and result in an empty field, resulting in the record not being written. If you wish to eliminate a record if the input field is totally empty, but include it if the input field contains spaces, simply uncheck the **Trim Blanks** attribute when you check the **Skip if Empty** attribute for that field.

The **Skip if Used** attribute is the opposite to **Skip if Empty** and omits records where the field is **not** empty. This may be used to produce two out files from one input file, separated based on the presence or absence of contents in an output field.

Selection of input records

Selection of input records Sometimes you might want to select or "filter" the input based on the contents of one of the fields in the input record. You can do this by checking the **Select** box on the **Misc** tab and entering some text into the edit box which

appears. While still on the **Misc** tab, click on the input field in which this text will be sought. When you do this, only records with that text in that field will be translated to the output file.

Appending Results to an Existing File.

If the **Append** box on the Output tab is checked, ParseRat will not overwrite an existing output file. Instead it will append the new information to the end of the old file. *The user is responsible for ensuring that the structure of the old file is appropriate and matches the new information.*

This function is useful for creating new records by splitting old ones. For example, if an input file contains names and addresses with several name fields for different people sharing an address (or if names are in "splittable" fields) a run may be done with **one** of the names going to a name field in the output record. After that pass, a different incoming name field may be directed to that same output field (setting "skip if empty" to avoid null names creating a record), checking the Append box and repeating the output to the same file. In this manner, one record will be created for each name.

Interleaving Output

Frequently you will have a need to print "several up". In other words, you may wish to use your output file as input to a mailmerge program which prints several mailing pieces on a single sheet of paper. If you have a lot of records and you want to maintain the same sequence as your input records after cutting up the sheets, this can result in frustration. It would be easier if you could just cut the stack of paper and put the resulting stacks together.

For example, if you have 2,000 records and you want to print 2 up on 1,000 sheets of paper you will want to print records 1 and 1001 on the first sheet then 2, 1002 on the second sheet. After cutting the stack the top piece would have records 1-1000 and the second stack 1001-2000.

To achieve this, you would use the "Interleaf Factor" setting on the Mailing tab and select the entry for "2 streams". This will order your output records as 1, 1001, 2, 1002, 3, 1003 etc.

Adding Header and Trailer blocks to the output file

You may insert a "block" of material ahead of the output records generated by ParseRat and/or a block of material behind the output records generated by ParseRat. The blocks of material are set up as files and those files are selected by entering their names on the **MISC** tab. If a full path name is given (or selected by the **Browse** button), then that file is used. If only the file name is given, then ParseRat will look for it in the directory in which the program is installed. If no file name is given, or the file cannot be found, then nothing will be inserted.

Link pages for web sites (automatically generated) A good example of how this is used is the automatic generation of HTML pages of links to web sites. You may set up a header block containing the HTML material to go ahead of the links and a trailer block (often consisting solely of `</BODY></HTML>`) to follow them. The other HTML code for the links is set up as default values in fields defined as empty (that is how constant information is placed in an output file) and the variable information for the URL, the clickable text and other variable material is selected from the source or the parser generated fields. Note that default values are limited to 31 characters, so a longer tag will need to go into more than one field. For link page generation, you should also select the "no delimiter" output format option.

As an example of this, you may look at an example created from the sample file **products.txt** (in the **samples** directory/folder). This is a comma-delimited file containing the name and description of a product together with a URL (web address) where it may be found. It looks like:

```
"Product" , "Description" , "URL"
```

```
"PlanBee project management planning tool 1.0c","Critical Path Project  
Management Planning tool. Inexpensive alternative to MS Project. Reports, PERT  
charts and Gantt charts may be either printed, or copied to the clipboard for  
pasting into other programs." , "http://www.guysoftware.com/planbee.htm"
```

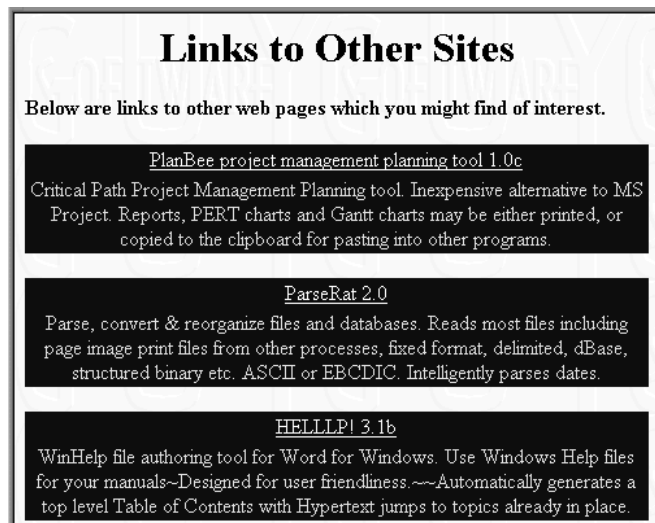
```
"ParseRat 2.0","Parse, convert & reorganize files and databases. Reads most  
files including page image print files from other processes, fixed format,  
delimited, dBase, structured binary etc. ASCII or EBCDIC. Intelligently parses  
dates." , "http://www.guysoftware.com/parserat.htm"
```

"HELLLP! 3.1b", "WinHelp file authoring tool for Word for Windows. Use Windows Help files for your manuals~Designed for user friendliness.~~Automatically generates a top level Table of Contents with Hypertext jumps to topics already in place.", "http://www.guysoftware.com/helllp.html"

If you open **products.txt** in ParseRat, ParseRat will recognize it as comma-delimited and present it as such. You may notice that the first record consists of field names, and check the "Take First Records as Field Names" box. To see how to create an output file with header and trailer block, load the saved preferences file **ProductToLinksPage.PRZ**.

Notice on the **Misc** tab that file names for a header block and a trailer block are included on that tab. Since they do not include full path names, they will be sought in the same directory/folder as the ParseRat program files (the installer will have put them there).

You may now save the output file (giving it an HTM extension) and it will look like the following when opened in a browser.



To create similar link pages from other input files (of any input format type), simply open the file and set it up so that the fields for the URL, clickable title and non-clickable descriptive text are available either as input fields or as Parser Generated fields. Then select **File... Copy Output Format** and select **ProductToLinksPage.PRZ**. Connect the appropriate fields to the output fields **URL**, **CLICKABLE TITLE** and **DESCRIPTIVE TEXT**. The fields **html1** etc. contain web page formatting tags which you should only change if you understand these tags.

Output the results to a text file with the HTM extension and open it in your browser.

Eliminating Duplicate Records

The elimination of duplicate records can be a major task in many situations.

It is not made any easier by the fact that records which in fact duplicate the same information are not identical.

Using mailing list examples, the name may be spelled differently (McDonald, Macdonald, MacDonald); the salutation (Mrs, Ms; Miss, Dr can be the same person); the street address (#123 - 456 17th Street; 456 Seventeenth St., Apt 123; Suite 123 seventeenth).

Commonly, when generating mailing lists from telephone list CD-ROMs duplicate records will also arise when the same entity has several listings (note that ParseRat has the ability to delete noise words and phrases like "Teen Phone", "Fax Line", etc.)

ParseRat has the ability to fully parse an address into its components, as well as generating a standardized address. It is also able to generate soundex coding of words.

If you know that the records which you wish to eliminate as duplicates are adjacent to those you wish to retain, the procedure is as simple as ensuring that those components on which you wish to base the "matches" are included among the output fields you specify on the Output tab, along with the records you wish to retain in your final file (you can remove the components you don't need later with another run of ParseRat).

Check the DeDup box for the match fields. For example, in a mailing list example you will probably want to create a soundex code of the last name (possibly of the street name if you think it might be mis-spelled but phonetically correct)

To obtain these components you may need to use the name parsing and address parsing features of ParseRat. You would check the DeDup box for the last name soundex, suite number, street number, street name soundex, city and state.

ParseRat would then include only the first record if it finds multiple records where those fields are identical.

If the records are not known to be adjacent, several passes are necessary.

Pass 1: Create the fields needed in an intermediate file.

Pass 2: Sort the intermediate file using the match records as keys (key sequence is not critical - this is just to bring them together). If you do not have a suitable sort program to handle the kind of output your later process needs (most will work with dBase and will sort tables), you might consider creating a fixed format record with no delimiters, placing the match fields at the front and using the DOS sort - you can process it later with ParseRat's fixed format routine.

Pass 3: Pass the sorted file through ParseRat with the "match fields" set as DeDup.

Pass 4: (Optional) Another pass to remove any match fields not required in the final output.

This process can be used with very large files (million record size certainly) although the sort may take a long time (maybe over night).

Saving Preferences

ParseRat has the ability (on the **File** menu) to save the settings you have made for a given situation.

You may wish to use them regularly for processing similar files (e.g. the monthly "hit" report you get from your WWW host).

These preferences may be saved (with the default extension .PRZ) for later use. Then when you are processing that type of file again, open the file then load your saved preferences for that type of file.

You may set many personal preferences with ParseRat, including preferred date output formats, modifications to standard lists of names, titles, designators etc which you will wish to use for many tasks on different file formats. You might wish to save them **before** you set input and output options under a name like **DEFAULT.PRZ** to save you time when setting up new file formats. Just open the new file, then load your saved default preferences, set the file-specific options and then you may save the resulting preferences under a new name.

You may also copy the output specifications from an existing preference file by selecting the Copy Output Format selection on the File menu.

Command Line Interface

Sometimes you might want to call ParseRat from some other program to carry out some pre-programmed task. You may do this by adding parameters to the command line when calling ParseRat. The command line interface is:

Parserat InputFileName PreferenceFileName OutputFileName

If all parameters are given, ParseRat will load the input file, load the preferences, write the results to the output file and then terminate, closing itself.

If you wish to read input from or place input to the clipboard use an asterisk * instead of a filename in the appropriate place.

If the preferences include "append" as an output preference, the new results will be added to the end of the output file if it already exists. If the preferences do not include "append", any file with the same name as the output file will be overwritten.

For example, if you regularly download or otherwise a file with a given name, you may set up a Preferences file to handle it. The results could be added to a database which you are keeping. A simple way of doing this would be to set up a batch file, say RegProc.BAT:

"Parserat" "download.txt" "regproc.prz" "results.dbf"
Exit

(The quotation marks are optional unless there are spaces in the file name and/or path).

And then associate this file with a desktop shortcut icon. Then, whenever you clicked on the icon, ParseRat would update your **results.dbf** file with information from **download.txt** following the rules you had saved in **regproc.prz**.

Using a Visual Basic or other program to call ParseRat via **shell** calls provides the means for some very sophisticated processing.

Note: If the output file name is omitted, ParseRat will load the input file, load the preferences and then wait for you specify an output file. If both output file and preferences are omitted, ParseRat will load the input file and wait for you to give it further instructions.

Hints and Tips

Favourite Settings: You might have settings changes which you *always* make. For example you might have names for which you always want to provide overrides to the gender data defined in ParseRat, or you might want to add titles or name suffices to the standards (or delete some that are there). The easy way to do this is to define these settings, without doing any output definitions and to save the resulting preferences. Then , when you start any new definition, just load that basic preference file as a starting point.

Multiple Passes:

Frequently a sophisticated parsing scheme may be obtained by using **multiple runs**, parsing first to a temporary file and subsequently using that file for input (see the suggestions for eliminating duplicate records using ParseRat). Another example may occur when parsing Canadian mailing lists. The Canadian postal code consists of two parts of the form V7V 1P9, etc. where the first part (the V7V here) indicates a broad tract called the "Forward Sortation Area". To extract the FSA from the postal code is simple if there is a field in the original file containing only the postal code (use the Split tab to break it into two fields at the space). However, if the code is combined in a field with city and province this won't work. To resolve this, you would first parse to an intermediate file to extract the postal code and then parse from that to your final file, splitting the postal code on the second run.

Data Blocks in Files:

You can do a lot more with the "Page Input" format if you consider pages to be "data blocks", not necessarily physical pages. Many files have data blocks logically separated by one or more blank lines. Process them by setting page input format with "blank lines" set as the page delimiter.

Tagged Input:

Consider using "tagged pages" to extract data from HTML files. Remember that a "page" could be a single line. For example, you might have an HTML file containing blocks like:

```
<INPUT TYPE=hidden NAME="ajparam_logpickord" VALUE="1">
Who holds political office in the city of
<SELECT NAME="ajparam_list1">
<OPTION SELECTED VALUE="0">Anchorage, AK
<OPTION VALUE="1">Fairbanks, AK
<OPTION VALUE="2">Juneau, AK
</SELECT>
and how can I contact them? <input type=submit value="Answer"></form>
</b></font>
```

If you wanted to pick out the "City/State" pairs as records, you would:

Select Page Input Format.

Select Tag Text on Top Line

Enter **<OPTION** in the tag text box (replacing the default "Page").

Select Data Location Tagged.

Go to the Page tab.

Click Add Tag.

Enter **>** in the box and press enter.

Note that a field name of **>** appears (the field name is the same as the tag) with a value of "Anchorage, AK". This may be further processed by such things as the "split" filter.

Pressing **next** produces a new record with **Fairbanks, AK** and pressing it again produces **Juneau, AK**.

Pressing **next** at this point will run down the file to the next such block and continue with more similar pairs.

Note that if the tag had been entered as **VALUE=** the field would have contained "0">Anchorage, AK and this could be split using the **Split** tab into three fields of 0, Anchorage and AK for output.

Using tags in this manner provides many sophisticated possibilities.

Examples in the Samples directory/folder

When ParseRat was installed on your system, a folder containing sample files was established.

This folder contains sample parsable files (ending in **.TXT**) and saved preference files (ending in **.PRZ**).

To access and examine these files, open the TXT file first, then choose the PRZ file from the **Load Saved Preferences** option on the **File** menu.

PHONEBK.TXT: is a **Comma Delimited with Quoted Strings** file as exported from one of the popular telephone list CD-ROMs. Open it and then load its saved preferences. It demonstrates several features.

Click on the **Next** button twice to move to record # 3.

Select the Output tab and look at the "Parser Generated Field" list to see how it has created several fields which you may use in manipulating names and addresses. (Note that the **Output** speed button at the top of the screen is disabled until the Output tab has been selected - this is to avoid having someone attempting to save an output file before defining fields to go into it).

Now select the Name tab. Note that the name is carried from "Field 1" of the parsed input line onto the name tab (in this instance the name is all in one of the input fields - ParseRat can assemble the name from several input fields if needed - usually to re-parse incorrectly parsed name components in the input file).

Notice that ParseRat has been set to expect input items in the sequence Last Name, First Name, Middle Names, Suffix and then Title and that it will create a "Standard" name in the sequence Title, First Name, Middle Names, Last Name, Suffix.

Titles and Suffixes will be recognized only if they appear in the lists given and that you may edit those lists. To add a tile, click on **Add**. To delete a title, click on it then click on **Delete**. Notice that you may provide substitutes to be used replacing titles and suffixes found (for example, you may wish ParseRat to recognize "Professor" as a title but replace it with -substitute - "Prof").

Notice also that in the raw name input there are items which are not appropriate for personal mailing purposes and which have been eliminated (click the Mailing tab to view the list of words which will be eliminated from incoming names - you may edit this list and your changes will be saved if you save preferences).

Notice that the title Mr has been generated although it did not occur in the input line. This results from the **Infer Title** checkbox being set. Uncheck the box and note that the title vanishes. ParseRat knows the gender for 5,000 names. You may set the preferred title for male and female names, you may also provide a title to use if both male and female names are present (usually denoting a couple). Many names are ambiguous (Kim, Robin etc. may be male or female). You may add or override the gender associated with a name (up to 99 changes or additions may be made).

Click three more times on **Next** and note how the title Dr was correctly identified after removal of some noise words.

Click **Next** until you reach record # 10, then click the **Street** tab.

Notice that this is an apartment number, although without an apartment designator. Notice that ParseRat has parsed out the address into components and has created an Odd/Even field in the Parser Generated Fields. See also how a "Standard" address has been created with the element sequence set according to the blue radio buttons.

ParseRat can provide a default suite designator for situations in which one is not found. Change the # in the default box to **Suite** and click the tab key. Note how the standard address now uses the new default.

Click **Next** until you reach record 17 and select the Name tab. Notice how the last name is parsed as two words, instead of the computer taking just the single word **De**. Multi word names are specified on the Mailing tab, where one or two word prefixes with are to be considered part of the surname are specified.

Click **Next** until you reach record 24. Note that **Van Der** has been parsed as the last name. Move to the Mailing tab and click on **Add** below name prefixes. Type in Van Der and go back to the name tab to see how the three word name has been recognized correctly (Van Der is a default prefixes in ParseRat - it was deleted from the saved preferences for this example for demonstration purposes).

LABELS.TXT: is a file typical of the output from a program which generates mailing labels. You might not have the means to do a direct export of the data, but can usually direct printed output to a file, which ParseRat can read. When you load the saved preferences for this file, you will see that it is set up as a "Page Image" file with a fixed length of 4 lines. The top three lines (the fourth is always blank) are specified to be individual fields. By setting the tabs well past the data on the line, each line is forced to be a complete field.

The first line is directly transferred to fields on the output tab. The second line is the "Street Address" portion, which is passed through the "Street" tab, with the "Standard Address" being output as the second field of the output record. The third line consists of city/state/zip information which for demonstration purposes is being parsed out into individual elements. Note that non-US zip information is also parsed out.

To see how the street address is parsed, select the Street tab and observe the results as you move through the file with the **Next** and **Prev** buttons. The sequence for the standard address elements is set using the block of blue radio buttons. The standardized abbreviations are defined on the **Mailing** tab.

PAGESMP1: is a page image sample, showing how several data blocks may be parsed from a single page.

The preferences loaded set it up as a page image file, with page breaks indicated by formfeeds (ASCII 12).

18 records result from this two page sample. Note how information from the header and footer is used in the output record as well as data from the data blocks.

PAGESMP2: is another page image sample. This one is typical of a report produced by some other program, from which data must be extracted. Note that it too has the page end indicated by FormFeeds (ASCII 12). The header area is set above the top data line and the footer area below the bottom data line. Note that this example has single row data bands and only one set of data across the page. Each data row will thus result in one output record.

Notice that the input data is passed through the Date and Time tabs, resulting in an output record in which the input date format differs from the output format. Note also that Year, Month, Day, Hour and Minute are also supplied in integer form as Parser Generated Fields.

As an experiment, select the Input Type tab and click the radio button for Tag Text On Top Line as a page indicator. Change the word Page in the edit box which appears to read Time (include the capital, these must match exactly) and return to the Page tab. Notice how the pages are now set up such that the word Time has now been used to select the page break position and anything above its first occurrence has been ignored.

Upgrades & Fixes
Any upgrades, maintenance releases and frequently asked questions will be posted at
<http://www.guysoftware.com/parserat.htm>

Please check the program help file for additional information which might not be in this printed manual.