ReadRPW

A program to read, summarise and export data from an Access query according to a sequence of instructions. Whereas most report writing tools (such as Access or Crystal Reports) read through a RecordSet and can group it according to specified hierarchy rules or code patterns, this program lets you specify the particular values to be allocated to each row. Ad hoc collections of values can be added to any row and they don't need to follow a regular sequence.

Other Report Applications	ReadRPW script
Detail …. CODE1 to CODE5	Row 1 …. CODE1 + CODE4 WHERE GROUP=1
Sub Total on GROUP 1	Row 2 …. CODE2 + CODE3 + CODE5
	Sub total …. Row1 + Row2

The risk of omitted or duplicated values is much greater with this type of report and the output must be regularly checked – or reconciling lines added to the report.

A script can be written with instructions about the titles, values, totals and print output. Each program line must be saved with a unique number (a sort order). These numbers can be renumbered (with steps or increments of 10) in order to assist with editing and inserting instructions into the script.

There may not be a need to place lines in the script sorted by the report row numbers. It is however essential that rows are defined in the script before any row which refers to them as a SUM for subtotalling. The sorting of the report output is determined by the print instructions in the Define_Output section.

There is only a very basic level of formatting available for the output (Titles, Skip Rows and Bold Rows). This tool is not designed to incorporate number formatting, fonts, borders or shading.

The Access program places the results in a table (RepOutput) which can be * used as a basis for other reports; * copied and pasted into Excel; * read by a program to create an HTML file.

Take care to use the correct formatting with square brackets and speech marks. Other characters such as round brackets or single apostrophes are not appropriate substitutes.

The report script must be in three sections, each with the appropriate heading.

Keywords: Define_Report, Define_Rows, Define_Output

Define_Report

Keywords: SOURCE, COLUMNS, END_COLUMNS, SAVEAS, EXPORTLOG

SOURCE: Name of the Access Recordset (**query**) which contains the data

COLUMNS: Each line must be in the format **Col*99* [FieldName] "Your description"** . The Columns must be in the range Col01 … Col20.

Note it **<u>is not</u>** possible to have calculated columns (e.g. Col01 = Budget, Col02 = Expend, Col03 = Col01 minus Col02). In order to achieve this, put the calculation into the source Recordset.

END_COLUMNS: Marks the end of the Columns section

SAVEAS: Optional – a filename which can be used for an HTML report file

EXPORTLOG: Optional – a filename which can be used to create a detailed log file (not yet available)

Define_Rows

Keywords: ROW, TITLE, PLUS, MINUS, SUM

Each row which contains values must be allocated a row number.

TITLE: A narrative to explain the row (Max 140 chars)

PLUS / MINUS: Add or subtract the values of the specified codes. A (slightly simplified) SQL criteria statement is used, containing field names from the specified source query.

MyRepDataQuery					
DIRECT	DEPART	GLCODE	DS1P	DS1C	DS4P
FINANCE	FINMNGT	AG4020PRID0	586	458	958
FINANCE	FINMNGT	AG4020PRIN0	764	130	518
FINANCE	FINMNGT	AG4020CONA0	452	963	500
FINANCE	FINMNGT	AG4020MSEB0	576	623	692

- PLUS [DIRECT]=”FINANCE”
- PLUS [GLCODE]=”******PRI**” WHERE [DEPART]=”FINMNGT”

SUM: For totalling rows which have already been populated with data values (PLUS or MINUS). It can also include mathematical symbols (+-/*) and real numbers.

A row can contain multiple lines specifying the content

- ROW 20
- PLUS [GLCODE] “AG4020PRIN0”
- MINUS [GLCODE] “******CON**”
- SUM [10]*0.5

It is not necessary to print every row in the Output section.

Define_Output

Keywords: TITLE, PRINT, BOLD, SKIP

TITLE: A row description without any values

SKIP: Leave a blank line in the report output

PRINT: Output a line and values to the report. A range of lines can be specified using an exclamation mark e.g. [100]![140].

BOLD: Same as PRINT but the font will be emphasised if output to HTML.

Note that a row can only be printed once in this section. If a row is required in different places in a report, define duplicate rows (with different row numbers) and SUM the value of the first row in the second

- 1000 Total Pay Sum [300]![900]
- 1010 Total Pay Sum [1000]

The script can specify the ordering of rows. The row numbers do not have to be sequential.

ProgText	ProgLine
define_report	1
source "MyRepDataQuery"	2
Columns	3
col01 [DS1P] "Current Period Expend"	4
col02 [DS1C] "Cumulative Expend"	5
col03 [DS4P] "Current Period Budget"	6
col04 [DS4C] "Cumulative Budget"	7
col05 [VAR41P] "Current Period Variance"	8
end_columns	9
	10
saveas "c:\mydata\myrpw.txt"	11
exportlog "c:\mydata\myrpwgllog.txt"	12
	15
define_Rows	16
row 8	17
title "dirfin consult"	18
minus [depart]="DIRFIN" and [GLCODE] like "??????CON??"	19
row 9	20
title "finmngt consult"	21
minus [depart]="FINMNGT" and [GLCODE] like "??????CON??"	22
	23
row 10	24
title "Dirfin spend excl Consultancy"	25
plus [depart]="DIRFIN"	26
minus [glcode]="AF1510MESM0"	27
row 20	29
title "FinMngt spend excl Consultancy"	30
plus [depart]="FINMNGT"	31
plus [glcode]="AF1510MESM0"	32
row 30	34
title "Finance Consultancy"	35
sum [8]+[9]	36
row 40	38
title "Total Finance"	39
sum [10]![30]	40
	41
define_output	42
title "Table1 Finance directorate"	43
print [10]![40]	44
Skip	48
title "Notes"	49
print [8]	50
Bold [9]	51
title "End of report"	52