R&R ReportWorks™

"Dynamic List ParameteRRs"

Strategies for Creation, Maintenance and Presentation of Table-Driven, Runtime Variable Value Selection for Reporting

A White Paper From Liveware Publishing

By Daniel Levin

Liveware® Publishing

History of End-User Prompting for Report Variables

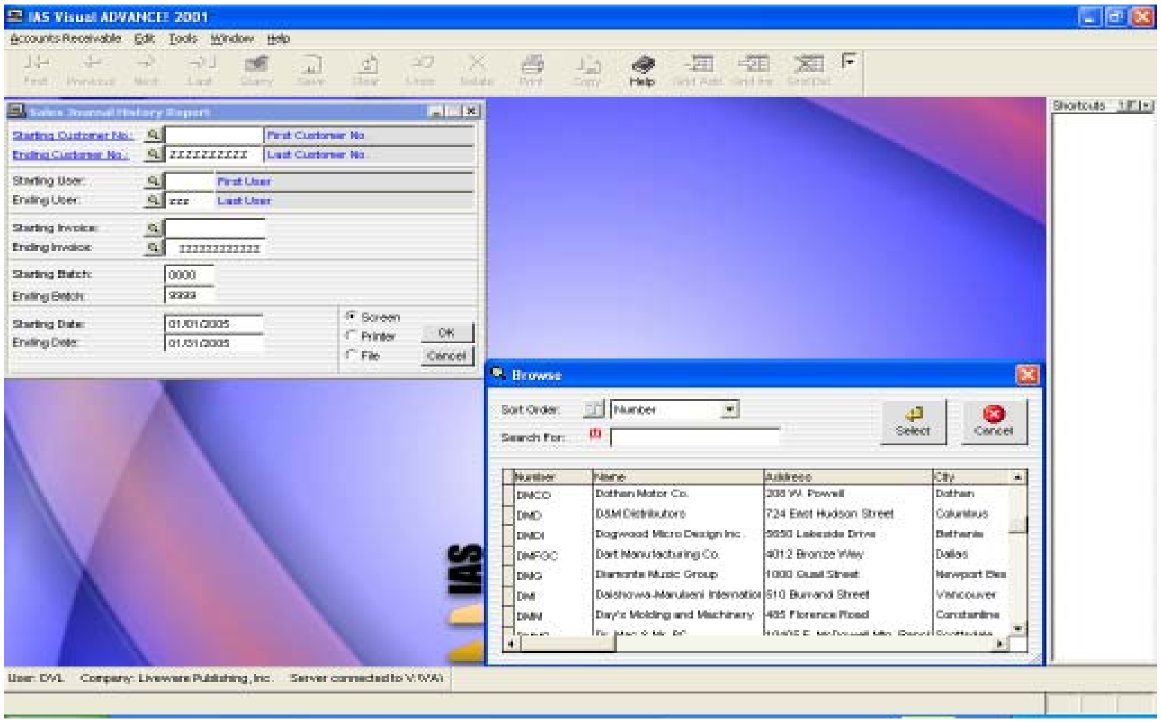
Beginning with R&R v9.0 released in April of 2001, Liveware has sought to expand R&R functional ability to present runtime entry variables to the end-user. Within R&R v9.0, Liveware introduced “ParameteRRs” – constant values that an end-user could specify whenever the report was generated, in order to control various aspects of the report’s generation.

Typically, ParameteRR entries controlled the query/filter, but they could also impact formulas, control sorting and grouping, linking, or any other item where a constant value – which would vary from one report generation occurrence to the next – was required. ParameteRR could contain any base R&R data type, including character, numeric, date, date/time or logical value. The report’s designer could restrict the allowable entries by virtue of a validation expression that must be true based on the end-user’s entry at runtime. In addition, R&R would control the data type entry, and present the entry based on the designer’s formatting selections.

The initial ParameteRR methodology introduced in R&R v9.0 successfully dealt with many of the most common runtime entry requirements. Users could now enter, for example, start and end dates to control the report’s generation. Still missing, however, were two very common input options. The first is a static, pre-defined list of available options; this is where the selection each option would result in a significantly different result at runtime. Version 9.0 ParameteRR could be made to mimic this kind of static list by using a combination of the text instructions and the validation formula to allow only certain entries. This methodology was difficult and only worked with a very short set of available options (at least based on the limitations of the text instructions). With the release of R&R ReportWorks in January of 2005, Liveware addressed this limitation by allowing an actual list of available, pre-defined options. Each option had its own description, and the report's designer could decide the order of the items in the list and a default value.

With the implementation of static-list ParameteRRs, only one category of commonly prompted runtime inputs remained: lists based on values in the database, a.k.a. a "dynamic" list. Such lists might either be too long or too subject to change (or both) to be pre-defined in a static ParameteRR. For example, a list of customer IDs would be changing regularly and may include hundreds or thousands of possible selections. Up through the present, Liveware has recommended the creation of a ParameteRR to hold the end-user input value for such an item, and instruct the designer to provide instructions to the end-user on how to lookup the correct value in the application itself. Admittedly, this was hardly satisfactory, but it was the best advice we could offer at the time.

Where a dynamic list was essential to effective runtime control, many application developers would choose one of two options. Some would opt to eschew a report writer in favor of defining reports within the application programming language itself. (See Figure 1.) Other developers (those that use R&R) would place within the application an entry screen that presents to the end-user the possible values from the database. Then the developers would send the selections to R&R via the RIPARAM() function, or via links to a DBF report control table that holds the selected values. (R&R Report Librarian uses this latter technique for some of its runtime reports.)



Starting Customer No.:		First Customer No
Ending Customer No.:	ZZZZZZZZZZ	Last Customer No
Starting User:		First User
Ending User:	zzz	Last User
Starting Invoice:		
Ending Invoice:	ZZZZZZZZZZZZ	
Starting Batch:	0000	
Ending Batch:	9999	
Starting Date:	01/01/2005	
Ending Date:	01/01/2005	

[✓ Screen] Printer File

Sort Order: Number

Search For:

Number	Name	Address	City
DMCO	Dothan Motor Co.	308 W. Powell	Dothan
DMD	D&M Distributors	724 East Hudson Street	Columbus
DMDI	Dogwood Micro Design Inc.	5650 Lakeside Drive	Bethania
DMFGC	Dart Manufacturing Co.	4012 Bronze Way	Dallas
DMG	Diamante Music Group	1000 Quail Street	Newport Bea
DMI	Daishowa-Marubeni Internatio	510 Burrard Street	Vancouver
DMM	Day's Molding and Machinery	485 Florence Road	Constantine

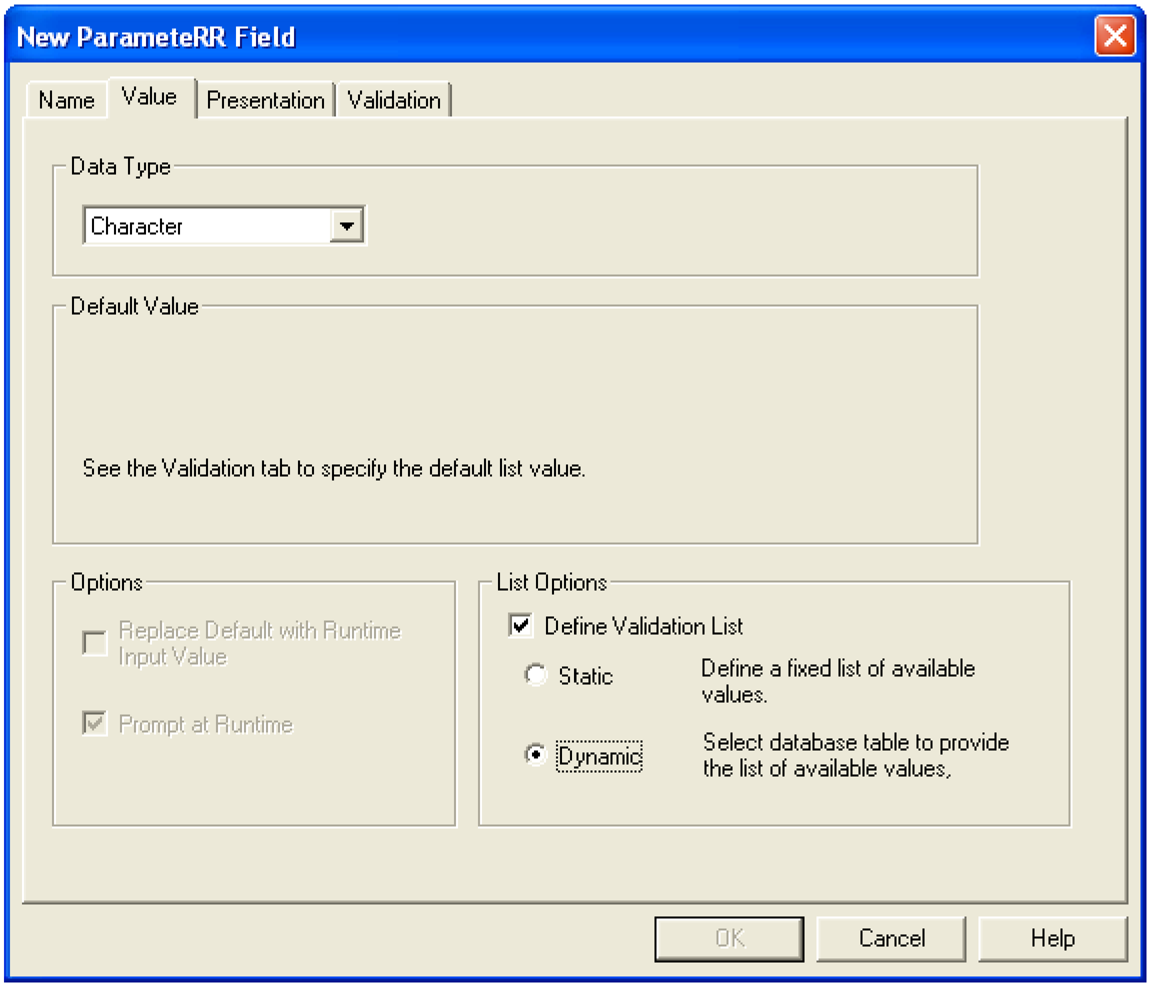
User: DVL Company: Liveware Publishing, Inc. Server connected to V:\VA\

Figure 1 – Example of user input from a list of database values, from IAS Accounting.

For a number of obvious reasons, Liveware does not like the idea that developers would choose *not* to use R&R for reporting due to dynamic list limitations. However, we also do not like the notion that a developer would have to build a special input screen to run a report. Therefore, we have designed R&R's dynamic, table-driven ParameteRRs so that the entire end-user interface to capture runtime inputs is specified within the R&R report itself.

Dynamic List ParameteRRs in R&R Report Designer

Beginning with the R&R ReportWorks release for April 2005 (Report Designer and Runtime modules v11.1), report designers may specify the new type of ParameteRR which is based on a .DBF table's contents. Those contents are entirely within the control of the report's designer; R&R will simply read and present to the end-user the table's records as they are stored in the .DBF file. Then the end-user will select the input value from the designated column from the table.



Data Type: Character

Default Value

See the Validation tab to specify the default list value.

Options

☐ Replace Default with Runtime Input Value

☑ Prompt at Runtime

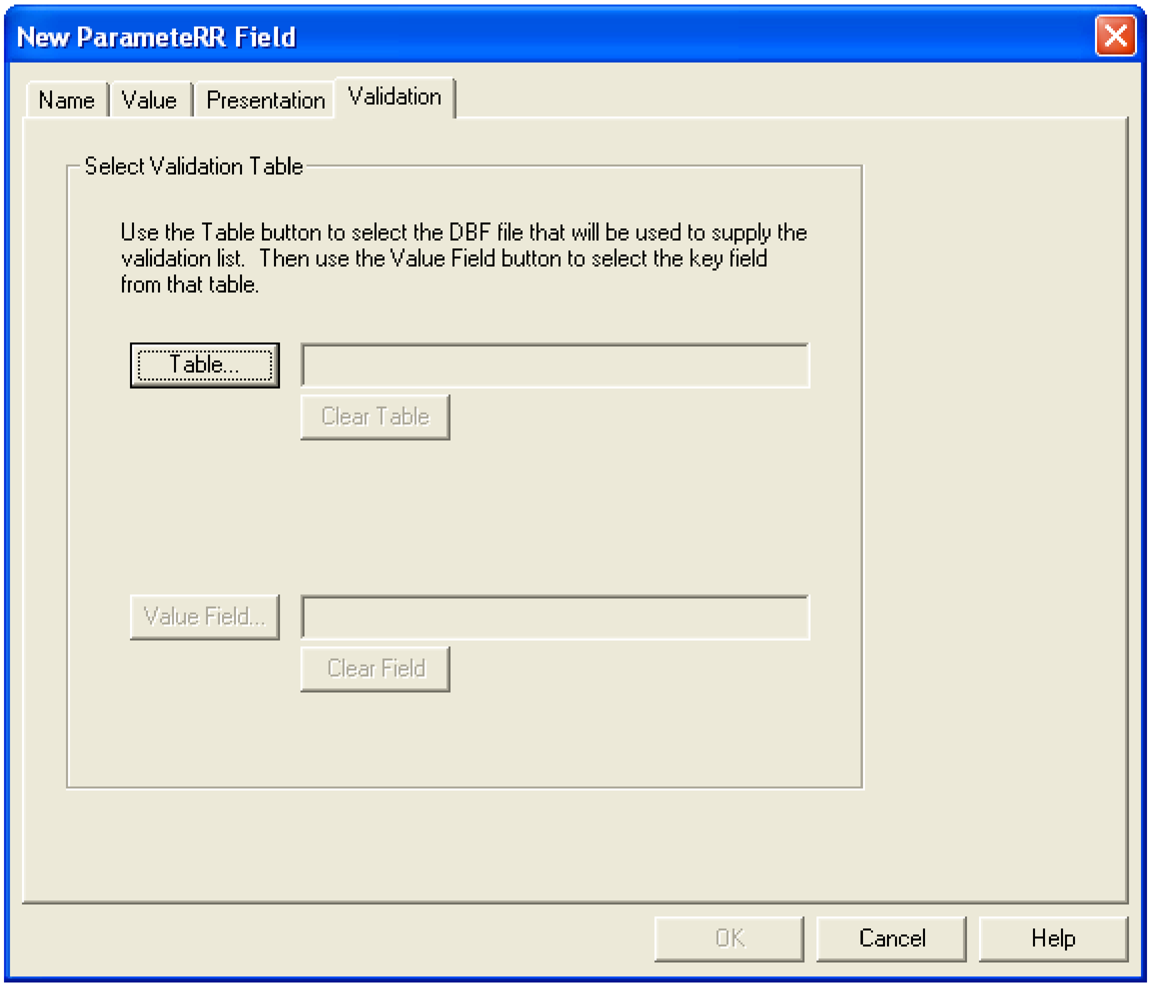
List Options

☑ Define Validation List

○ Static — Define a fixed list of available values.

● Dynamic — Select database table to provide the list of available values.

Figure 2 – ParameteRR "Value" tab with Static/Dynamic List options


New ParameteRR Field

Select Validation Table

Use the Table button to select the DBF file that will be used to supply the validation list. Then use the Value Field button to select the key field from that table.

Figure 3 – ParameteRR "Validation" tab for Dynamic List specifications

Within the ParameteRR definition dialog, we have added a new selection on the "Value" tab. When the designer checks the option to define a validation list, he or she will be presented with the option for either a "Static" or "Dynamic" list. (See Figure 2 at left.) The "Static" lists are all existing list-based ParameteRRs, and they will be unchanged.

The designer's selection of a "Dynamic" list will cause R&R to offer new options on the "Validation" tab (shown above). The designer will then specify the .DBF file that will contain the records to be presented to the end-user at runtime. Once that table is selected, the designer then indicates which field (column) of the target table contains the values that will be returned to R&R to serve the ParameteRR's value. R&R will only list the fields of the data type specified on the "Value" tab; changing the data type on that tab will invalidate the prior field selection.

Once specified, the Dynamic ParameteRR's field can be used in any valid R&R context based on data type. Other contents of the .DBF are only present to assist the end-user in selecting the desired value from among the options presented in the target data field.

Creation of the .DBF File for a Dynamic ParameteRR

Since the R&R will only present the .DBF file, the designer should take sufficient care in its creation so that it presents the end-user with information that assist in the selection of the ultimate ParameteRR value. Liveware has opted to require .DBF files – as opposed to other table formats – for several reasons. First, .DBF is a non-proprietary file structure so no specially licensed software is required to create them. Second, .DBF files can be easily updated or overwritten without undue database management overhead. Third, .DBF files already contain column names which R&R will display as column headers. Fourth, .DBF files include data fields or, essentially, the same data types as R&R would use in its various contexts: character, numeric, data, date/time, logical.[1] Fifth, R&R already contains all of the code to read .DBF files for regular reporting purposes. Sixth, R&R can itself create the .DBF files that its own reports may need to use.

R&R SQL Report Designer and Runtime will also use .DBF files to populate Dynamic ParameteRRs, even though these tools are designed to read other types of database tables. For reasons described below, this is still the most efficient way to provide for table-driven ParameteRR selections, even in the SQL environment.

Any table will contain a list of fields of various data types. The simplest table contains just one field; in the context of a Dynamic ParameteRR such a table would list only the available values. While such a

[1] Dynamic ParameteRRs for logical fields are not applicable, since the only two options are TRUE and FALSE. This is hardly meaningful in a dynamic context.

CUSTCODE	CUSTNAME	CITY	STATE
AAC	Allied Appliance Co.	Dearborn	MI
AALP	American Alpine Supply	Boulder	CO
ABK	Albert's Books	Ithaca	NY
ACCD	Accounting Dynamics	Athens	GA
ACO	Alfredson Company	Wilmington	NC
ADY	Adylford's Florist	Bakersfield	CA
ALRI	Alarm Results, Inc.	Princeton	NJ
AMMD	Amarillo Motors & Drivers	Amarillo	TX
AMX	American Expresss Corp	New York	NY
APR	Assoc. Product Mfg.	Gainesville	FL
AUI	Alabama United, Inc.	Huntsville	AL
AVLE	August Von Lear Co.	Pittsburgh	PA
AWI	Advanced Watches Inc.	Evanston	IL
AWIB	Advanced Watches (Balto)	Baltimore	MD
BAE	William A. Early Inc.	Jackson	MS
BBI	Balanced Breakfast Inc.	Gary	IN
BCLD	Birmingham Closet Distrib.	Birmingham	AL
BCM	Berringer Climate	Seattle	WA

Figure 4 – Simple table with customer codes and descriptions

table would work within R&R, most of the time the table value passed to the R&R ParameteRR will be a coded character or numeric value representing that record. For example, Figure 4 displays a typical Customer database, where the customer code column contains the values that will be passed to R&R. While the code values are all that R&R cares about, the end-user, in order to make the correct selection, would need more information than simply the codes. Certainly, the customer's name and location would assist the end-user's selection. While this sample table includes just code, name, city and state, the report's designer must decide what additional columns will be needed in the table.

The next example illustrates this point in a bright light. Suppose, instead of the customer code, the user must select an invoice number from a long list of available entries. For such a need the designer should probably include the invoice date, the customer code and customer name, invoice status, and total invoiced amount. (See Figure 5.) Even under those circumstances, that information might not be enough since all of the fields shown could contain identical reference information for two or more records. (In this circumstance, the same customer could order the same amount twice on the same day.) Therefore, the .DBF file presented to the end-user many have to contain many more columns than one might initially suspect.

Figure 5 also illustrates another critical aspect of table-driven ParameteRR value selection. In most modern software applications, relational tables will contain only information sufficient for storage within the table. The table containing invoice header information (such as the invoice number, date, amount and status) would not typically contain the

INV_NUM	CUSTCODE	CUSTNAME	INV_DT	STAT	INVAMT
29380	KRD	Kramer Dynamics Co.	01/12/05	O	1250.37
29432	JJE	John J. Early, Inc.	01/17/05	O	39283.45
29543	ABK	Albert's Books	01/19/05	P	283.00
29544	ABK	Albert's Books	01/19/05	P	283.00
29594	ILQ	Int'l Liquids	01/19/05	O	29837.28
29596	DJEL	Elvis DJ Entertainers	01/19/05	O	293.73
29597	WAS	Washington Supply	01/19/05	I	387.54
29599	DUE	Duluth Environmental	01/19/05	O	3827.47
29600	OLWR	Oliver Warehousing	01/19/05	P	570.00
29604	LIVA	Liverpool Vanities	01/19/05	O	981.25
29606	UNTH	United Theaters	01/19/05	P	45.29
29607	APR	Assoc. Product Mfg.	01/19/05	O	1093.50
29609	RED	Redding Muffler	01/19/05	O	283.09
29610	GRN	Granger Pharmacy	01/19/05	O	94.26
29611	GRN	Granger Pharmacy	01/19/05	O	104.27
29613	UNT2	United Theaters II	01/19/05	P	538.30
29614	EFR	Efficiency Radiation	01/19/05	C	9371.36
29615	SEY	Southeast Yard Maint.	01/19/05	O	847.99

Figure 5 – Sample table with multiple invoice data columns

CID	CCTY	CST	CDESC
03299	Carlsbad	NM	Regal Automotive, Inc.
19277	Ames	IA	Whitaker Hospitality
21736	Palmdale	CA	Phillips Service Station
33309	Petersburg	VA	Hansen Accountants
38274	Dover	NJ	Landscaping & More
43382	St. Paul	MN	Sherman Consulting Group
47210	Tampa	FL	General Brokerage Inc.
55201	Fresno	CA	Fresno Tire
68832	Topeka	KS	Rollins Trucking
72341	Staten Island	NY	Richmond Dry Cleaners
88734	Hampton	VA	Towne Liquor Mart
94421	Watkins Glen	NY	Travelers Aid & Support
94832	Boise	ID	Miller Services Co.
95802	Columbus	OH	Berger, Howard PC
99382	Portland	OR	Portland Paving, Inc.

CUSTNUM	CUST_NAME	CITY	STATE
95802	Berger, Howard PC	Columbus	OH
55201	Fresno Tire	Fresno	CA
47210	General Brokerage Inc.	Tampa	FL
33309	Hansen Accountants	Petersburg	VA
38274	Landscaping & More	Dover	NJ
94832	Miller Services Co.	Boise	ID
21736	Phillips Service Station	Palmdale	CA
99382	Portland Paving, Inc.	Portland	OR
03299	Regal Automotive, Inc.	Carlsbad	NM
72341	Richmond Dry Cleaners	Staten Island	NY
68832	Rollins Trucking	Topeka	KS
43382	Sherman Consulting Group	St. Paul	MN
94421	Travelers Aid & Support	Watkins Glen	NY
88734	Towne Liquor Mart	Hampton	VA
19277	Whitaker Hospitality	Ames	IA

Figure 6 – Generic data table (top) versus specially prepared data table (bottom)

customer's name, which is a key piece of information needed for the end-user to select the desired invoice record. Therefore, simply having R&R read and present an application's stored table would rarely suffice in the context of a table-driven ParameteRR. The prompting table will almost always have to be generated in advance of the report.

All tables contain a set of records in a particular order. Further, each record contains field entries, each field has a name, and fields appear in some order, Each of these table elements is important to the context of assisting an end-user to find the record containing the correct value to transfer to the ParameteRR. Figure 6 demonstrates the issues that ensue when some of these elements are ignored or misapplied. In the sample table on the top of Figure 6, the records are sorted by the first column, the customer code. This will make the table difficult to use in the context of selecting those codes, because the end-user will most likely know the customer’s name, not their numeric code. The bottom table correctly sorts the records by customer name, making the task of locating the desired customer code much simpler.

In Figure 6, there are other advantages of the bottom table. The column headers are descriptive, in place of the generic field names from the source application found in the top table. The customer name is also immediately to the right of the customer code, rather than two columns over. As Figure 6 illustrates, the table’s contents, field names, field order, record order, and record selection are all important at various times for various table-driven ParameteRRs. In order to make the entire effort work – and worthwhile – the designer must be able to build the .DBF file from the ground up.

Liveware’s software engineers and designers discussed this issue at length, and analyzed several possible methodologies that might assist a report’s designer to create the .DBF file for a table-driven, Dynamic ParameteRR. As the analysis progressed, we first determined that we might offer the designer a dialog box within the Dynamic ParameteRR definition screen with options necessary to define the .DBF file. In that way, the R&R report could contain all the instructions necessary to build the prompting .DBF file. We ultimately rejected such a design – at least for now – as too restricting. We found that this dialog would have to contain, essentially, all of the R&R functions necessary to design a report, except for the layout. That includes calculated and total fields, sorting, grouping, filtering, etc.

The other limitation we found disturbing was that R&R would have to rebuild the .DBF files every time it ran the report, which may cause undue delay in the processing of the main report. If the .DBF file already existing that would serve the table-driven ParameteRR, R&R should just be able to present it.

Based on these two considerations, we decided that the report’s designer would specify the .DBF file for R&R to present to the end-user. R&R would simply present whatever data was in that file, and leave it to the report’s designer to insure that the .DBF file contained what he or she intended.

SLSCODE	SLS_NAME	TERRITORY	STATUS
P3984	Williams, Joseph P	Pacific	ACTIVE
N0357	Stevens, Monica	Northeast	ACTIVE
S9388	Abdul, Hakim	Southest	ACTIVE
M1193	Jenkins, Walter R	Mountain	TERMED
M3847	Taylor, Rachel G	Mountain	ACTIVE
N0498	Rotelli, Anthony J	Northeast	ACTIVE
L9928	Polchek, Andre	Lakes	LEAVE
T3401	Busch, Randall	Texas	ACTIVE
P2901	Hardinger, Joan	Pacific	ACTIVE
P4387	Richards, Dick R	Pacific	TERMED
A4833	Thompson, William	Atlantic	TERMED
N3912	Stringer, Becky A	Northeast	ACTIVE
A0948	Dole, Libby	Atlantic	LEAVE
L2883	Travis, James A	Lakes	ACTIVE
S9228	Reynolds, Alexander	Southeast	ACTIVE
S1128	Tasker, Robert J	Southeast	LEAVE
A9011	Patterson, Linda m	Southeast	XFERD
T7711	Sanchez, Jose	Texas	ACTIVE

Figure 7 – Sample table with salesperson data

Strategies for Preparing the .DBF File for Dynamic ParameteRRs

The final design described above was not the culmination of our analysis, however. We knew that for table-drive ParameteRRs to be effective, we had to offer strategies for the report's designer to create the necessary .DBF files containing the input and prompting values, and to have the file refreshed when needed. Fortunately, R&R itself could perform any of the data processing functions to create or refresh the .DBF file. R&R could do so by means of a separate report specification that would create the target .DBF file, whether from the SQL or xBase report designer. Further, report designers could use any other tool that produces .DBF files (as most data management programs do) if they so chose. [2]

We will review in some detail, below, considerations when building the .DBF file using either R&R or some other program (although we will demonstrate the issues using R&R's Report Designer). But first, we must address the critical issue of timing the creation or refresh of the .DBF file.

As described above, R&R at runtime will present the .DBF file for table-driven ParameteRRs as it exists at that time. How often that .DBF file will need to be refreshed is an issue the report's designer must address at

[2] Some may question why we did not adopt XML as the standard format for table-driven ParameteRRs, instead of .DBF. XML, they would argue, is becoming the standard for data interchange. We do not agree with that assessment, and XML has disadvantages of .DBF files for structured data files. XML requires lengthy text tags, which are often many times larger in space consumed than the data presented. Use of XML format would require the resulting data file to be that much larger and to take that much longer to rebuild.

design-time. That answer is driven primarily on the context of how the report will be used. More specifically, the question to ask is: just how “dynamic” does this Dynamic ParameteRR need to be.

A couple of examples can illustrate this point. Figure 7 shows the table needed to allow an end-user to select a salesperson code for a prospective monthly commission report. The party running the report would, most likely, be a sales department manager or clerk who would know the salespeople, if not always remember their codes. The salesperson table in the application software would be the source of the table-driven ParameteRR information, but direct use of that table would not make sense since it would likely contain dozens of fields, most of which are of no use to the person running the report. The only information the sales manager or clerk would care about are the salesperson code (to pass to the ParameteRR), name (to correctly ID the salesperson in question) and, perhaps, current status. Better still, it would be *ideal* if the table contained *only* salesperson records for whom the manager or clerk might wish to generate the monthly commission report.

If the context for the monthly commission report’s use is always exclusively the current month – as, for example, to attach to a commission check – then a monthly refresh of the .DBF file with just active salespeople would be ideal. Unless, of course, this report is used throughout the month and salespeople are added to the database all the time. Then the .DBF file for the table-driven ParameteRR might need refreshing each time the report is run.

That last extreme case would be the exception, however. Many reports will require that their table-driven ParameteRR be refreshed only occasionally. The life-cycle of the .DBF file will depend on several factors including:

1) How often are records *added* to the source tables?
2) How often is critical identity information for existing records changed?
3) Can including some additional information in the .DBF file negate the need to refresh the .DBF file as frequently?
4) How likely is it that the end-user will not be able to select the correct ParameteRR value from the .DBF file, either by selecting the wrong value because it *changed* or because it was *not included* in the .DBF file R&R presented?
5) How much confusion is caused by inclusion of records in the .DBF file that the end-user would never select?
6) Will inclusion of records that the end-user would never select result in a .DBF file that contains so many records that the end-user can’t easily locate the record they want?

There are no straight-forward answers to these questions and they require a knowledgeable balancing to reach the correct conclusion.

During our evaluation process, we debated all of these questions and decided that always refreshing the .DBF file for ParameteRR prompting was unrealistic. Such a methodology, while eliminating all of the above refresh timing issues, would cause undue delay in processing the report. Further, we determined – from our extensive[3] knowledge of applied reporting – that few reporting applications would require continual refresh of the table-driven ParameteRR data. And for those rare cases, R&R give developers the tools to refresh the .DBF file whenever it is necessary.

In the end, we decided that it was wiser to allow the report's designer or application developer to determine the method and timing of the .DBF file refresh by a contextual balancing of the factors above. To see how decision that might apply in the 'real world', we will return to our example salesperson commission report.

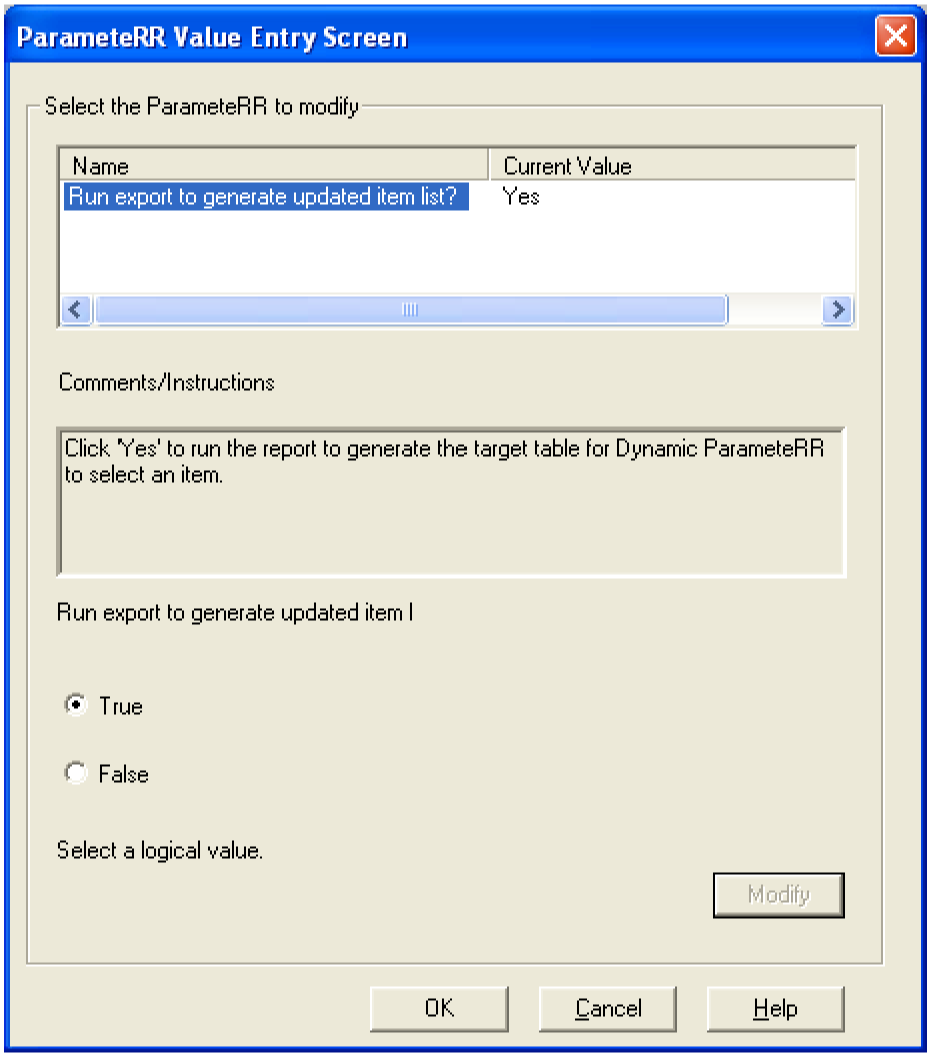
With just the information provided above, issues 1) and 4) are probably the most critical ones to address in this context. If the sales manager or clerk attempts to run the commission report and the company had added the desired salesperson since the last refresh of the .DBF file, then use of the report hits a wall. The report's designer could, however, offer an alternative ParameteRR to accept a value that was not listed in the .DBF file. The clerk may have to then look up the correct value in the application software.

On the other hand, if the company is large and adding dozens of salespeople all the time, the regular refresh – perhaps requiring up-to-the-minute .DBF file refresh – could be necessary to make the report truly useful. That would probably be the exception, however.

After any evaluation that balances the factors described above, the answer to the 'when-to-refresh' question will be either 'on-demand', 'periodically', or 'always'. R&R supports techniques for each answer, as we illustrate below.

We will start with the 'always' answer; that is, the target .DBF file must be refreshed each time the R&R report calling for it is run. In nearly every case where this is required, the R&R report would be activated via the Runtime module. A developer in this circumstance would have two options: create the .DBF file via their application program or some other tool or *first* run an R&R report that refreshes the Dynamic ParameteRR's target .DBF file. Since the developer would already be using R&R Runtime

[3] Liveware's design and development team includes a total of almost 50 years of cumulative, direct experience with reporting applications for hundreds of clients.


ParameteRR Value Entry Screen

Select the ParameteRR to modify

Name	Current Value
Run export to generate updated item list?	Yes

Comments/Instructions

Click 'Yes' to run the report to generate the target table for Dynamic ParameteRR to select an item.

Run export to generate updated item l

(●) True

( ) False

Select a logical value.

Figure 8 – ParameteRR entry screen that would control whether a Dynamic ParameteRR's data file would be refreshed or not

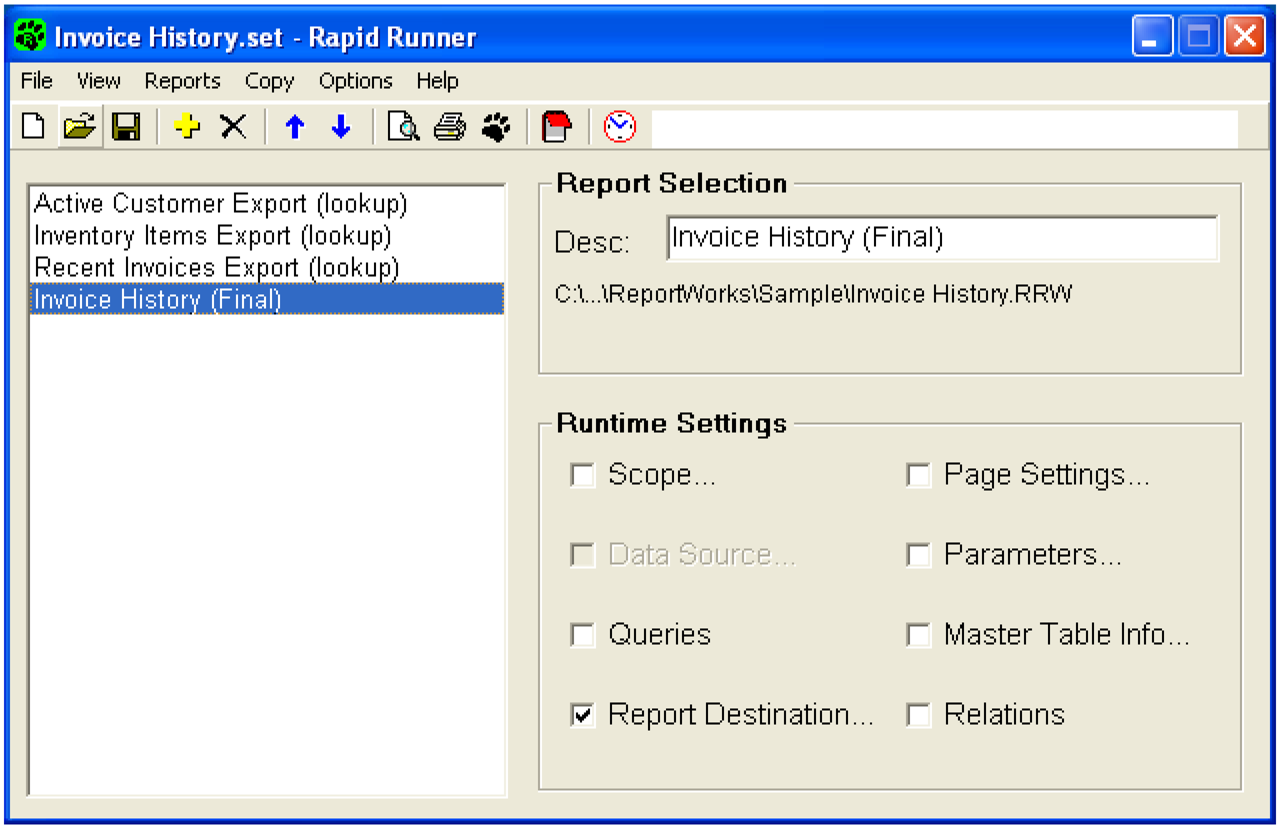
to generate the final report, he or she could instruct the application program calling Runtime to first generate the target .DBF file via an R&R report that builds the .DBF file in the folder location anticipated by the second report. In theory, if multiple target .DBF files were needed, the developer could run several R&R reports (or other programs) to create or refresh those target .DBF files.

While the approach described above could take some time to execute, and the end-user would have to wait until completion before making their final ParameteRR selections, the wise developer could a) include either a "wait" notice while the target .DBF files were being built, b) offer the end-user options as to whether to refresh all or a specific target .DBFs, or c) a combination of the two. While an R&R ParameteRR could present such an option (see Figure 8 above), we expect most developers would offer a programmatically-control screen to give end-users some information to assist them with the refresh decision, such as the date/time the target .DBF file was last modified or refreshed.

If the developer would rather NOT bother with the coding to present the end-user with the controls to rebuild the target .DBFs, R&R's Rapid Runner module can perform the same function. As shown in Figure 9, Rapid Runner is designed to run either a set of reports in sequence from a batch, or the end-user can also designate just those reports from the batch they wish to run. Since that is exactly the issue described above, Rapid Runner is the ideal tool for the job. (Note that a developer may launch a Rapid Runner report set from within an application program in the same way that they launch R&R Runtime.) Since the final report that calls the target .DBFs would need to run last, the developer would list that report last (as shown in Figure 9) and the end-user could simply select only that final report to run if they knew it was unnecessary to refresh any of the target .DBF files.

When a developer decides that only periodic refreshing the target .DBF files is warranted, the weight of programming needs diminishes greatly. Our experience is that most reporting applications would be well-served with periodic refreshing, whether the period is daily, weekly, monthly or even hourly.

When R&R reports are generated via Runtime called from an application program, a developer can build in commands to rebuild the



Active Customer Export (lookup)
Inventory Items Export (lookup)
Recent Invoices Export (lookup)
Invoice History (Final)

Report Selection

Desc: Invoice History (Final)

C:\...\ReportWorks\Sample\Invoice History.RRW

Runtime Settings

- [ ] Scope...
- [ ] Page Settings...
- [ ] Data Source...
- [ ] Parameters...
- [ ] Queries
- [ ] Master Table Info...
- [✓] Report Destination...
- [ ] Relations

Figure 9 – Rapid Runner report set with .DBF file rebuilds and final report

DEPTCODE	DEPT_DESC	DIVISION	HEADCT2005
1640-3	Accounting	ADMIN	17
2820-4	Advertising	SALES-MKTG	10
3740-2	Banding	PRODUCTION	37
2019-4	Bidding	SALES-MKTG	3
1827-2	Casting	PRODUCTION	12
0293-1	Executives	EXEC	4
4920-3	Facilities	ADMIN	6
2911-3	Human Resources	ADMIN	10
2810-2	In-line processing	PRODUCTION	56
1027-3	Logistics	ADMIN	5
2109-4	Marketing	SALES-MKTG	4
2019-2	Metalworking	PRODUCTION	25
3722-2	Packaging	PRODUCTION	9
1928-3	Purchasing	ADMIN	2
0293-3	Research and Development	ADMIN	4
0192-4	Sales	SALES-MKTG	14
4322-2	Shipping	PRODUCTION	6
2218-2	Warehouse	PRODUCTION	10

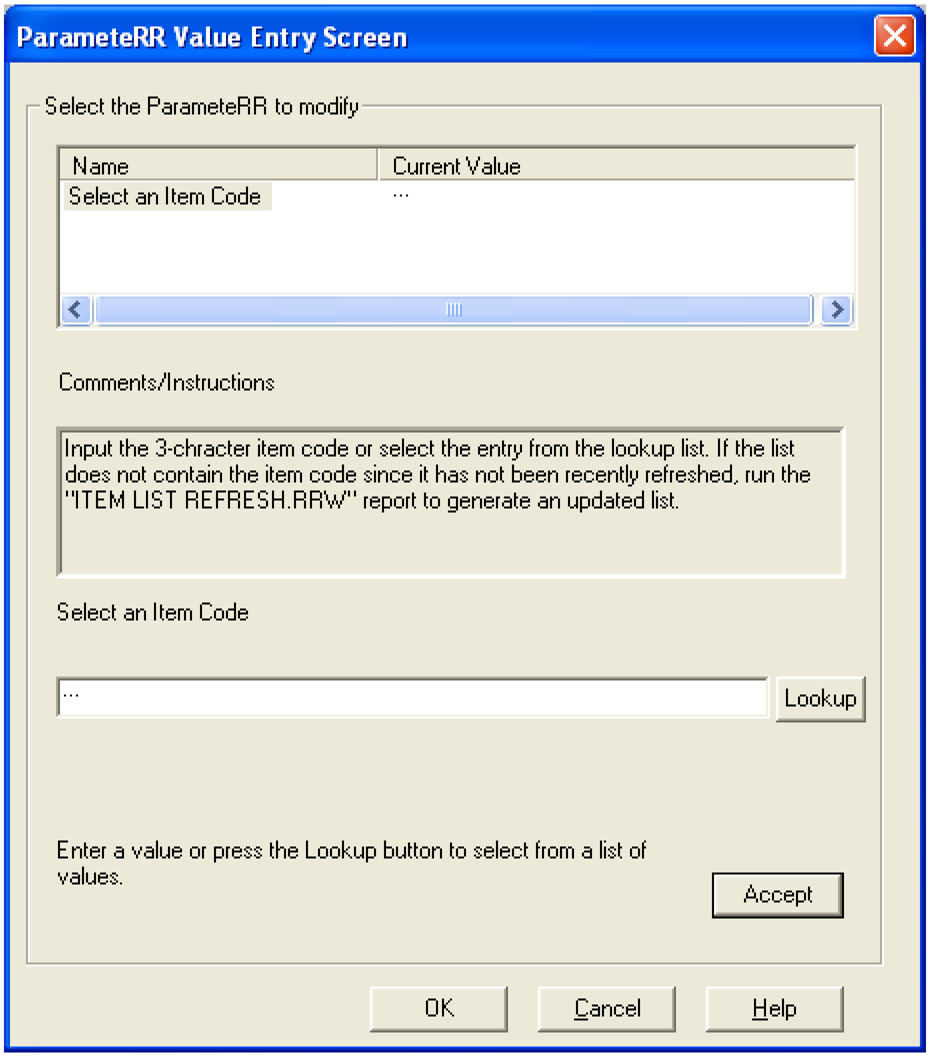
Figure 10 – Sample data table that would rarely require a data refresh

target .DBF files in the background. Another option is to add menu options to refresh batches of target .DBF files with recommended frequencies.

A programmatic approach is not the only one available. Application or reporting administrators could simply schedule themselves or other programs to launch a series of R&R reports that generate refreshed target .DBF files, or those same parties can create Rapid Runner batches that perform that maintenance function. (E.g., see Figure 9, but without the last report entry.) These target .DBF files could – and probably should – reside in special folders on the network, so that R&R can find them whenever an end-user needs to run a report. Since Rapid Runner includes a scheduler, these refresh activity can occur unattended; we suspect that's how most administrators will resolve to perform the function.

In effect, periodic refresh of these Dynamic ParameteRR target .DBF files produces a sort of data warehouse for custom reporting applications.[4] We have designed R&R so that the target .DBF files are only in use while an end-user is making a selection from within the window. Multiple parties could access the same target .DBF file simultaneously, but overwriting of an open file would, naturally, be disallowed. This is another good reason to schedule refresh of most of the target .DBF files for off-hours.

[4] We have discussed this approach at length in other white papers, including the one entitled "Desktop Data Warehousing" available on Liveware's web site at www.livewarepub.com/white_papers.htm.


ParameteRR Value Entry Screen
Select the ParameteRR to modify
Name
Current Value
Select an Item Code
...
Comments/Instructions

Input the 3-chracter item code or select the entry from the lookup list. If the list does not contain the item code since it has not been recently refreshed, run the "ITEM LIST REFRESH.RRW" report to generate an updated list.

Enter a value or press the Lookup button to select from a list of values.

Figure 11 – ParameteRR dialog with "Instructions" to execute a data refresh on the target table controlling the Dynamic ParameteRR

Certain target .DBF files would rarely need to be refreshed. For example, a report where the Dynamic ParameteRR would contain a department code calls for a target table that hardly ever changes. (See Figure 10.) In some cases, a Static ParameteRR may be appropriate, but if the list of departments is long or the designer wishes to display several pieces of data about the department (in addition to the code and description), a Dynamic ParameteRR serves the purpose better. In addition, the same target .DBF file can apply to multiple reports, while a Static ParameteRR would need to be created or changed in each report where it was defined.

The refresh for this target .DBF file is 'on-demand', in that there is little reason to update it unless circumstances warrant it. That would occur either when their was a documented change to the source data, or when an end-user could not access the department code needed for the Dynamic ParameteRR value because it was either not in the table or not clearly marked. (See factors 1 and 4 above.) To address these occurrences, we suggest that the instruction box that accompanies the ParameteRR entry contain a message similar to that should in Figure 11 (page 16).

R&R Runtime creates a separate window, so it should be possible to execute another program or R&R report that rebuilds the target .DBF file while keeping the original report's ParameteRR dialog active. If the application developer did not want to allow this, the end-user could always terminate the report, run the process to rebuild the target .DBF file, then execute the original report again.

There are a few other considerations for preparing and maintaining these target .DBF files. For most reporting applications the same values are needed to populate many Dynamic ParameteRRs. Customer code or department code would often be a value requested by many reports for many different purposes. While the contexts for making the selections may be slightly different, many times the same target .DBF file can be used on many reports simply be adding a little more information to those tables. Since the target .DBF files might be refreshed on a more aggressive schedule for some reporting needs than others, all reports that share the same target .DBF file could gain the advantage of the most aggressive refresh schedule for that table.

Further, the target .DBF file need not contain *only* data from the application's tables. In addition to deriving values from that data (including, for example, total YTD sales as shown in Figure 12), report designers can add reference text and define more explicit column headers in order to assist the end-user.

CUSTCODE	CUSTNAME	FYTD_SALE	NOTE1
AAC	Allied Appliance Co.	2920.13	Last Sale:03/13/05
AALP	American Alpine Supply	47738.02	Last Sale:11/24/04
ABK	Albert's Books	11284.88	Last Sale:10/10/04
ACCD	Accounting Dynamics	328.17	Last Sale:09/07/05
ACO	Alfredson Company	49003.40	Last Sale:02/19/05
ADY	Adylford's Florist	8374.29	Last Sale:12/31/04
ALRI	Alarm Results, Inc.	3772.00	Last Sale:11/29/04
AMMD	Amarillo Motors & Drivers	120.00	Last Sale:08/11/04
AMX	American Expresss Corp	0.00	Last Sale:02/17/03
APR	Assoc. Product Mfg.	2664.43	Last Sale:03/02/05
AUI	Alabama United, Inc.	993.25	Last Sale:04/30/04
AVLE	August Von Lear Co.	11827.23	Last Sale:07/13/04
AWI	Advanced Watches Inc.	29304.02	Last Sale:11/18/04
AWIB	Advanced Watches (Balto)	180.20	Last Sale:12/02/04
BAE	William A. Early Inc.	0.00	Last Sale:10/19/05
BBI	Balanced Breakfast Inc.	1283.23	Last Sale:10/03/04
BCLD	Birmingham Closet Distrib.	473.99	Last Sale:02/12/05
BCM	Berringer Climate	32827.10	Last Sale:01/08/05

Figure 12 – Customer information with YTD sales and other reference info

UNIQUE_KEY	CID	DIV	CDESC
03299NE	03299	NE	Regal Automotive, Inc.
19277MW	19277	MW	Whitaker Hospitality
19277SW	19277	SW	Whitaker Hospitality - AZ Region
19277WE	19277	WE	Whitaker Hospitality - CA Region
21736WE	21736	WE	Phillips Service Station
33309AT	33309	AT	Hansen Accountants
38274NE	38274	NE	Landscaping & More
43382MW	43382	MW	Sherman Consulting Group
47210SE	47210	SE	General Brokerage Inc.
47210WE	47210	WE	General Brokerage Inc.
55201WE	55201	WE	Fresno Tire
68832MW	68832	MW	Rollins Trucking
72341NE	72341	NE	Richmond Dry Cleaners
88734AT	88734	AT	Towne Liquor Mart
94421NE	94421	NE	Travelers Aid & Support
94832WE	94832	WE	Miller Services Co.
95802MW	95802	MW	Berger, Howard PC
99382MW	99382	MW	Portland Paving, Inc.

Figure 13 – Dynamic ParameteRR target table with a compound unique key, one made from two or more data field in the source table

At times, even the target .DBF file's column containing values to pass to the Dynamic ParameteRR may need to be derived. In some application programs, one must combine more than one field to determine a unique key, as demonstrated in Figure 13. Assuming consolidated reporting on several separate "divisional" databases, the customer code and division ID combine to form a unique value.

Presentation of Target Table to End-Users via R&R Runtime

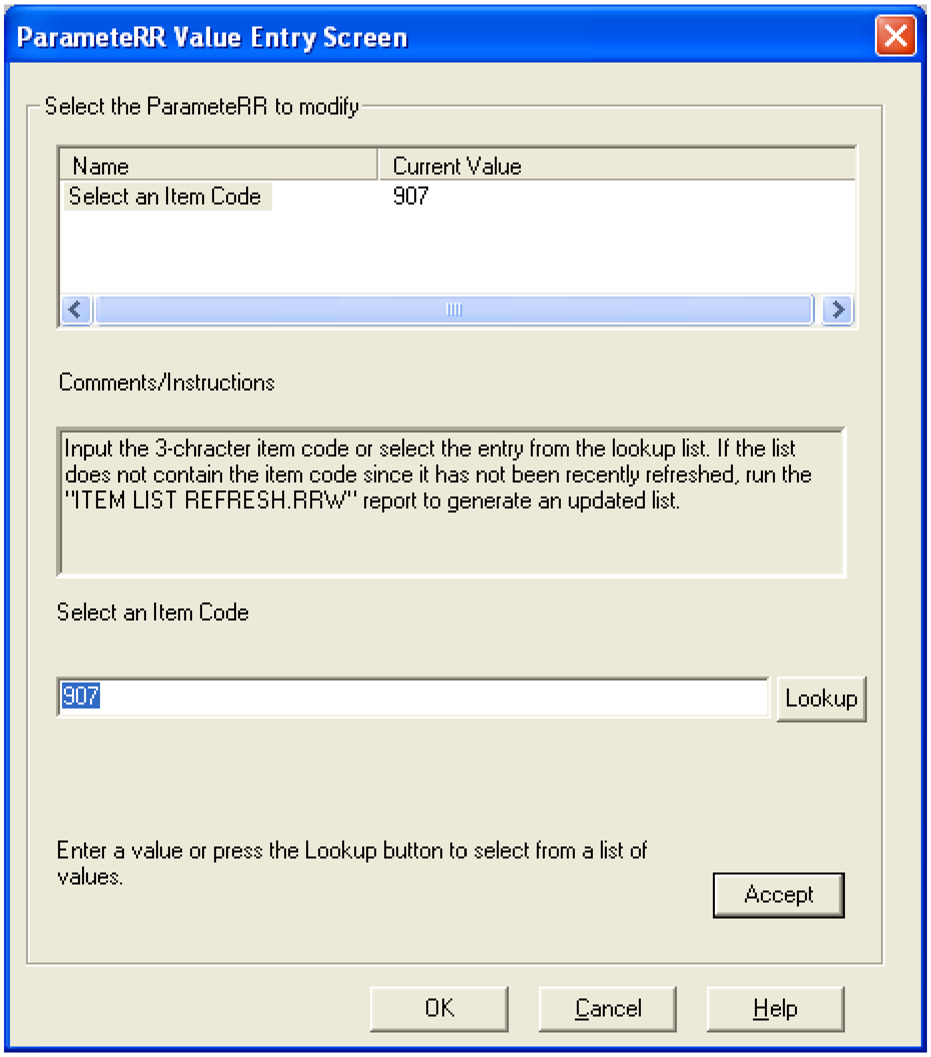
All of the report designer's and developer's efforts achieve fruition when an end-user activates an R&R Runtime report and is asked to populate the value for a Dynamic ParameteRR. As shown in Figure 14, Dynamic ParameteRRs are displayed in the list just like other ParameteRRs. When the end-user selects the Dynamic ParameteRR to assign a value (which will always default to null for the data type), the end-user may either input the value or click the icon adjacent to the entry box to view the target .DBF file. As with any other ParameteRR the report's designer may specify instructions to assist the end-user.

The report's designer defines, as described above, the path and .DBF file name within the report itself. When the end-user attempts to view the target file, R&R will first try to locate that file in the path stored within the

[5] In SQL reports, master table file path does not apply. Therefore, R&R would skip this step.

report. If R&R can't find the file in that folder, it will attempt to locate it in the folder for that report's master table.[5] If the target .DBF file is not in that folder, R&R will check the folder where the report itself resides. Lastly, after these unsuccessful location attempts, R&R will prompt the end-user to locate the .DBF file in another folder by offering the Windows file locator dialog. Once the user has specified a folder location for the target .DBF file, if the report has any other Dynamic ParameteRR R&R will search that user-selected folder prior to again presenting the Windows file locator dialog.

Report generation is not conditioned on the existence or ability to locate the target .DBF. The end-user can input a value or allow R&R to process the report with a null value for the ParameteRR. Therefore, unattended use of R&R runtime can continue even though the report contains a Dynamic ParameteRR. When designing reports that are intended for unattended generation (as, for example, to be used via Rapid Runner scheduling), one should avoid Dynamic ParameteRR or allow for valid operation with either an alternate null or default value.


ParameteRR Value Entry Screen

Select the ParameteRR to modify

Name	Current Value
Select an Item Code	907

Comments/Instructions

Input the 3-chracter item code or select the entry from the lookup list. If the list does not contain the item code since it has not been recently refreshed, run the "ITEM LIST REFRESH.RRW" report to generate an updated list.

Select an Item Code

907

Enter a value or press the Lookup button to select from a list of values.

Figure 14 – Dynamic ParameteRR selected in the ParameteRR entry dialog at runtime

When the end-user clicks to icon to display the target table, R&R will display a modal window with all of the information from the .DBF file, as shown in Figure 15. In the window header R&R will display the target file name and path, date and time created, and the ParameteRR name that will be populated. The end-user can resize the window to allow for a wider view of the data as their monitor allows. Standard Windows scroll bars are provided for both vertical and horizontal movement.

The left-most column of the table will always be the target field that contains the list of possible values for the ParameteRR. These values need not be unique; that is, the same value can appear multiple times within the table, if that's what is in the .DBF file. This column will always be locked on the left, so that even if the end-user scrolls right to view additional columns, the end-user will always be able to view the available values.

As previously explained the remainder of the table's fields will be displayed in the order they are stored the .DBF file. Similarly, records are listed in the order they reside in the .DBF file, so care should be taken in sorting of the records when building the .DBF file.[6]

[6] The code necessary to allow for manipulation of the .DBF file within the window would be extensive, and would dramatically increase the size of the R&R runtime executable. The increased executable file size would significantly slow down loading runtime, so we decided that this deficit outweighed the benefits.

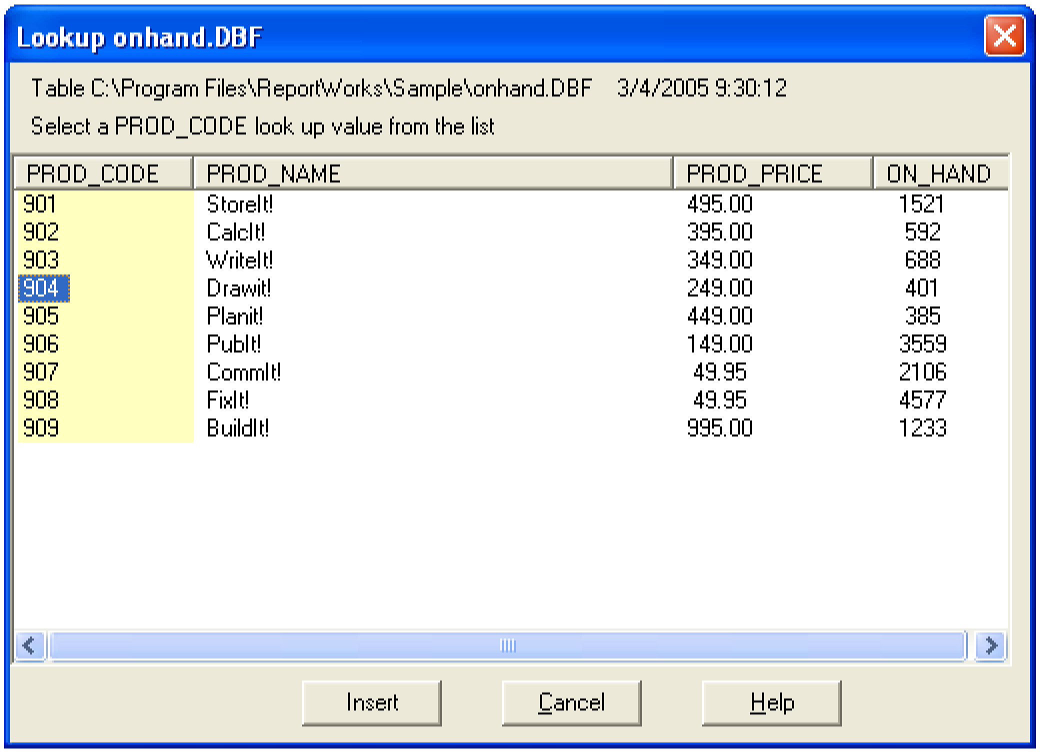

Lookup onhand.DBF

Table C:\Program Files\ReportWorks\Sample\onhand.DBF   3/4/2005 9:30:12

Select a PROD_CODE look up value from the list

PROD_CODE	PROD_NAME	PROD_PRICE	ON_HAND
901	StoreIt!	495.00	1521
902	CalcIt!	395.00	592
903	WriteIt!	349.00	688
904	DrawIt!	249.00	401
905	PlanIt!	449.00	385
906	PubIt!	149.00	3559
907	CommIt!	49.95	2106
908	FixIt!	49.95	4577
909	BuildIt!	995.00	1233

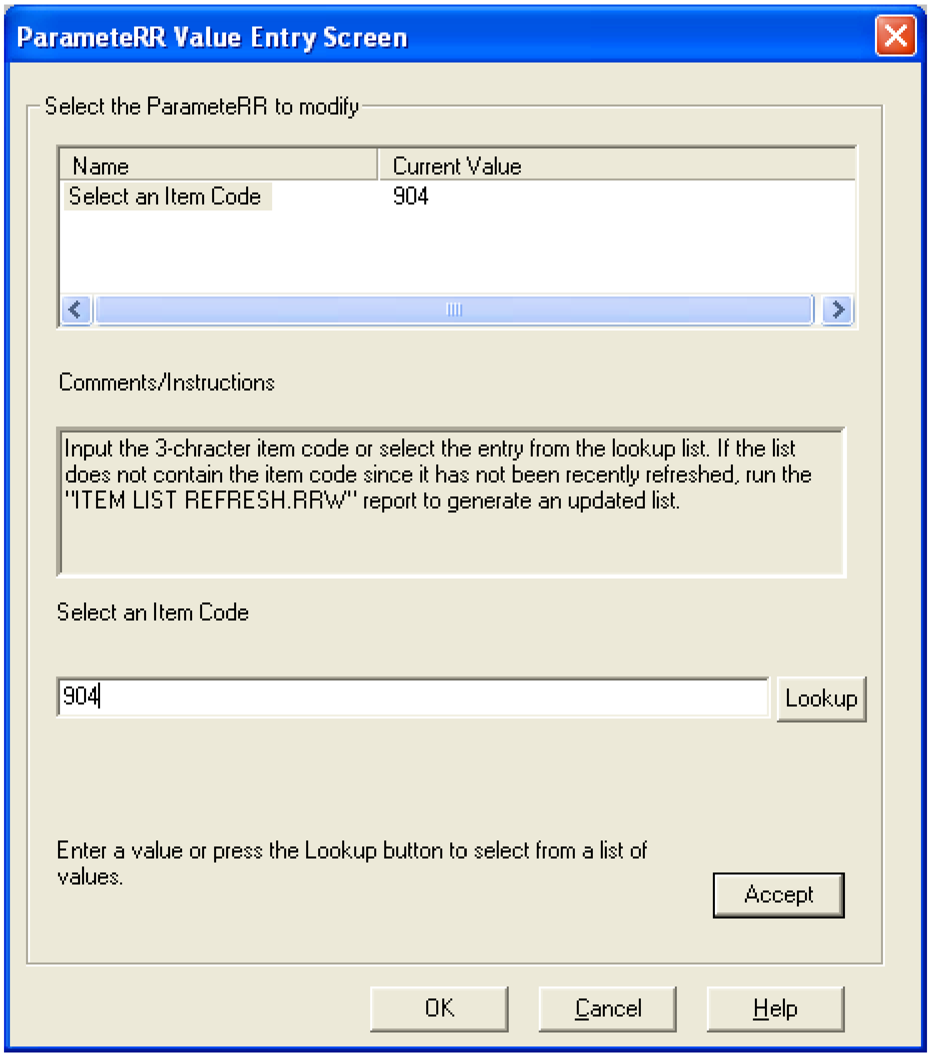
Figure 15 – Selection window for Dynamic ParameteRR value

To select the record containing the target column's value to populate the ParameteRR, the end-user simply clicks the mouse on the row of the table. The window reflects the selection with <TBD>. Then end-user can then hit <ENTER> or click the "Select" action button to confirm the selection. The end-user can, alternatively, hit <ESC> or click "Cancel" to return to the previous dialog.

Upon the end-user's selection of the desired value, the ParameteRR dialog will present the entry in the box as shown in Figure 16. The end-user can select a different value by again clicking the icon to present the .DBF file window, or delete or edit the value placed in the entry box.

Summary

Dynamic ParameteRRs offer report designers and system developer many new options to employ and deploy R&R reports. While some care must be taken to effectively implement this new feature, the positive returns are substantial.


ParameteRR Value Entry Screen
Select the ParameteRR to modify
Comments/Instructions
Lookup
Accept
OK
Cancel
Help

Name	Current Value
Select an Item Code	904

Input the 3-chracter item code or select the entry from the lookup list. If the list does not contain the item code since it has not been recently refreshed, run the "ITEM LIST REFRESH.RRW" report to generate an updated list.

Select an Item Code: 904

Enter a value or press the Lookup button to select from a list of values.

Figure 16 – Dynamic ParameteRR value selection transferred from data table window

Existing reports may be retrofitted to adapt Dynamic ParameteRRs, making the value of upgrading to the latest R&R version highly advantageous and worthwhile. For new report development, particularly within software applications, the savings in time and effort in designing a lead-in screen to gather variables could well be substantial. All existing methods of passing variables to R&R Runtime remain unchanged, including the RIPARAM() function and other ParameteRR categories.

In future R&R releases, we plan to add other functionality in this area, including the ability of an end-user to select multiple values from lists for both Static and Dynamic ParameteRRs.

Our technical support and consulting staff stand ready to assist anyone wishing to implement Dynamic ParameteRRs and is looking for advice.

About the Author and R&R ReportWorks

Daniel Levin is President of Liveware Publishing, Inc., which became the publisher of R&R Report Writer in September 1999. Mr. Levin, his partner, Christian A. Strasser, and associates have worked over the last decade and a half with hundreds of companies through their consulting practice to help those clients use their data most effectively. As an R&R Authorized Trainer beginning in 1993, Mr. Levin has taught over 100 reporting classes and published two books on R&R and the principles of database reporting: ***Relate & Report: Your Guide to Reporting with R&R*** (pub. 1996) and ***The R&R Cookbook*** (pub. 2000).

R&R ReportWorks superseded R&R Report Writer Version 10+ upon its release in January 2005. More information about R&R and other white papers and educational resources are available on Liveware Publishing's web site: www.livewarepub.com. R&R is sold throughout the world and has an estimated 500,000 users within tens of thousands of businesses, governments, non-profit organizations and educational institutions. R&R ReportWorks has a retail price of $600, covering Report Designer modules for xBase and SQL reports, Runtime engines and other utility modules. (Price at publication.)