Introducing

RattleR&R

Affordable Business Intelligence for the Web

Introduction (Overview)

Web based reporting has been a feature of R&R since Version 8 which added support for HTML export and for generation of PDI report output which can be read using an ActiveX control which is downloaded to a client browser. To use these features for real time web reporting application developers needed to write the interface to request information from the client and then launch R&R runtime on the web server to execute the report, send it to a file and deliver it to client browser.

RattleRR expands on this technology base by using an ASP COM component interface to deliver reports on the web. It is a direct web interface to the R&R runtime engine that requires minimal programming on the server and absolutely no client configuration.

RattleRR is a server-based tool that allows you to execute and communicate with your R&R reports using Active Server Pages (ASP) and an IE compatible scripting language.

Its foundation is the R&R Data Warehouse. Data Warehouse includes both the Xbase and SQL editions of R&R. As part of the RattleRR installation, Data Warehouse is installed on the web server and then the RattleRR specific components are installed and registered.

Using the Report Designer you develop the reports and then from there you simply create an ASP page where you set the name of the report to be run on the server along with any additional runtime properties and the page delivers the report output to the web client.

Installation and Configuration

RattleRR must be installed on a functioning web server that is also capable of running the R&R Data Warehouse. We have tested RattleRR using Personal Web Server under Windows 98, IIS4 under Windows NT and IIS5 under Windows 2000. Browser clients have included IE4 through IE6.

The RattleRR components must be placed in specific areas of your server and the server must be properly setup so that the scripts can be run, the registry is properly updated and you have a logical and consistent methodology for placing your reports, temporary data files and web application.

The RattleRR installation program allow you to select the install location for:

Xbase program directory	Location of RRRWRUN.EXE and RRWATL.DLL
SQL program directory	Location of RSRWRUN.EXE and RSWATL.DLL
RattleRR Sample directory	Location of R?SAMPLE.ASP examples
Report Control directory	Location of RRPRVIEW.CAB

You need to develop your reports so that the data will be available to the runtime executable and also need to ensure that any RattleRR ASP scripts are in a location on your web server having script execute privileges. RRPRVIEW.CAB needs to be installed where it can be located from within a RattleRR script file. Typically the root directory of your web server works best.

How RattleRR report features differ from Runtime

RattleRR allows you to use most of the functions (such as changing a master file or specifying a query condition) that are available via traditional runtime. However the following differences exist between using RattleRR and using traditional runtime to execute reports.

- RattleRR reports are always output to a browser report window where they can then be printed to a local printer.
- Reports stored in report library files are not supported by RattleRR.
- RattleRR does not support any of the file export options that are available via runtime.
- User prompt dialogs (such as displaying the interactive query dialog or using parameteRR fields) are NOT available in RattleRR.
- User input to reports must be done via the definition of RIPARAM() based calculated fields that can then be passed through RattleRR.
- The current default RattleRR behavior for scope and query/filter is to include ALL records rather than using saved report settings unless a value is explicitly set within the ASP page.
- For SQL reports, the report data source MUST be a System data source and NOT a user or file data source since the browser client does not come in to the server as an authenticated user.

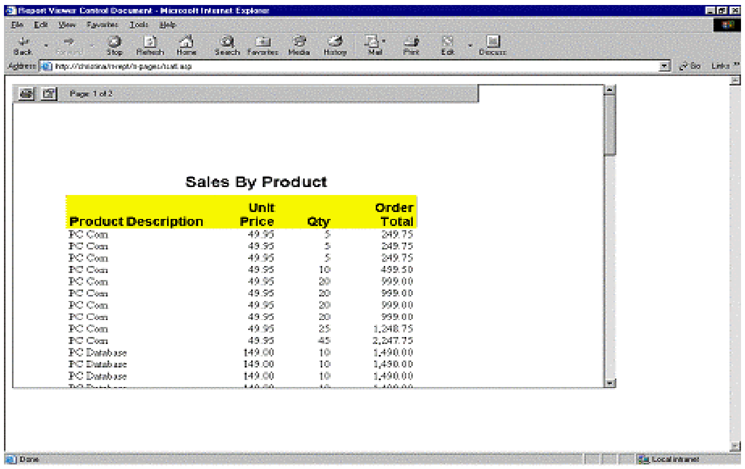
Using RattleRR to run reports

Running a report via RattleRR is really quite similar to running conventional R&R runtime. In both cases, you invoke a saved report with a particular set of criteria and then output the report results. For RattleRR the input is always done via an active server page and the output is always the delivery of an activeX report window within a browser window.

The syntax used within a RattleR ASP page is similar to what you'd see if you were generating runtime code using the ActiveX control. The major difference in using RattleRR is that the control exists directly within the server and not as an object embedded on a form within a program.

To use RattleRR, you create an ASP page that points to the report that you wish to run. When a browser opens the ASP page, it initializes the RattleRR server object, reads the report specific properties and passes information to the R&R runtime DLL. The runtime DLL then in turn calls the R&R runtime executable to process the report. The report output is sent to a temporary PDI file whose contents is then displayed in the report window on the browser using the Report Viewer Control that is downloaded and installed on the local machine from the RRPRVIEW.CAB that is present on the web server.

Here is an example of a report invoked by an ASP page displayed in a browser window. The report retains all of its formatting and can be scrolled and printed.



Report Viewer Control Document - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address
Go Links
Page 1 of 2
Done
Local intranet

Sales By Product

Product Description	Unit Price	Qty	Order Total
PC Com	49.95	5	249.75
PC Com	49.95	5	249.75
PC Com	49.95	5	249.75
PC Com	49.95	10	499.50
PC Com	49.95	20	999.00
PC Com	49.95	20	999.00
PC Com	49.95	20	999.00
PC Com	49.95	20	999.00
PC Com	49.95	25	1,248.75
PC Com	49.95	45	2,247.75
PC Database	149.00	10	1,490.00
PC Database	149.00	10	1,490.00
PC Database	149.00	10	1,490.00

We have included two sample ASP pages to get you started. RRSAMPLE.ASP (for Xbase version) and RSSAMPLE.ASP (for SQL version) provide the basic required elements to execute a report via RattleRR. Each of these sample scripts is included as part of the RattleRR installation.

You can use these files as the starting point for your own scripts that must be placed on the web server in a directory where script permissions have been enabled.

RattleRR Active Server Page Structure

A RattleRR ASP page requires a number of elements to be present in order to deliver a report to a browser window. There are also a number of optional elements to control aspects of the report as well as the overall appearance of the report viewer browser window.

Required Elements

The first required element within a RattleRR ASP is a CreateObject() call to the appropriate control. For the Xbase version the control is RRATL.RRWControl and for the SQL version it is RRATLRSW.Control. Once the object is created, you set the report specific properties of that object. Minimally you need to set reportName with a report specific name and runReport to execute that report. You can optionally introduce a variety of properties and methods to specify runtime options for the report such as a selection criteria or a sort order. Each available property and method is described later in this file.

The “else” clause of the ASP contains a number of important sections. The first is the path pointing to RRPRVIEW.CAB, the file containing the client-side ActiveX components that gets added to the browser. To enable the browser and force the download of the components, the URL path (not the local PC path) to this file relative to the location of the ASP must be specified. This is done in the section of the ASP where “codebase=” is specified. You should only change the path preceding RRPRVIEW.CAB and leave the remainder of the line just as it appears in sample page.

Adding additional formatting and HTML on the report page

The sample RattleRR pages deliver a browser screen with the report displayed with the report control window. You may want to embed additional HTML in your ASP pages so that more than just the report itself will be displayed. This could be to provide information relating to the report itself, instructions on printing, or to retain a consistent graphical look and feel related to your other pages.

You can edit the HTML returned within the “else” clause of your ASP by placing valid HTML tags and information within the your page. A simple example is shown below:

The title has been set to “The report you requested” and “Here’s Your Report!” has been placed in the body section embedded within the <H1> tags:

```
<HTML>
  <HEAD>
    <TITLE>The Report you requested</TITLE>
  </HEAD>
  <BODY>
    <H1>Here's Your Report!</H1>
    <object WIDTH="85%" HEIGHT="85%"
     . . .
```

Specifying the space allotted to the report window

You may want to control the amount of browser “real-estate” taken up by the report displayer. Modifying the percentages in the

    <object WIDTH=”85%” HEIGHT=”85%”

allows you to do this. These figures define the percentage of the browser window (height & width) that the report window will use.

RRSample.ASP

```
<%
if Request("generatereport") = "true" then
     Response.Buffer = TRUE ' For IIS 4 compatibility
     set rr = Server.CreateObject("RRATL.RRWControl")
     rr.reportName = "You must enter a valid report file name.RRW"
     rr.runReport
else
%>

<HTML><HEAD>
<TITLE>Report Viewer Control Document</TITLE>
</HEAD><BODY>

<object WIDTH="85%" HEIGHT="85%"
           CLASSID="CLSID:66960E23-DE25-11CF-876F-444553540000"
            codebase="rrprview.cab#Version=2,0,0,4" id=RepView1>
           <param NAME="LanguageID" VALUE="0409">
           <param NAME="ReportURL" VALUE="<%=
Request.ServerVariables("SCRIPT_NAME") %>?generatereport=true">

           <embed WIDTH="95%" HEIGHT="95%"
           CLASSID="CLSID:66960E23-DE25-11CF-876F-444553540000"
           CODEBASE="rrprview.cab#Version=2,0,0,4"
           TYPE="application/oleobject"
           PARAM_ReportURL=""></object>
</BODY>
</HTML>

<%
end if
%>
```

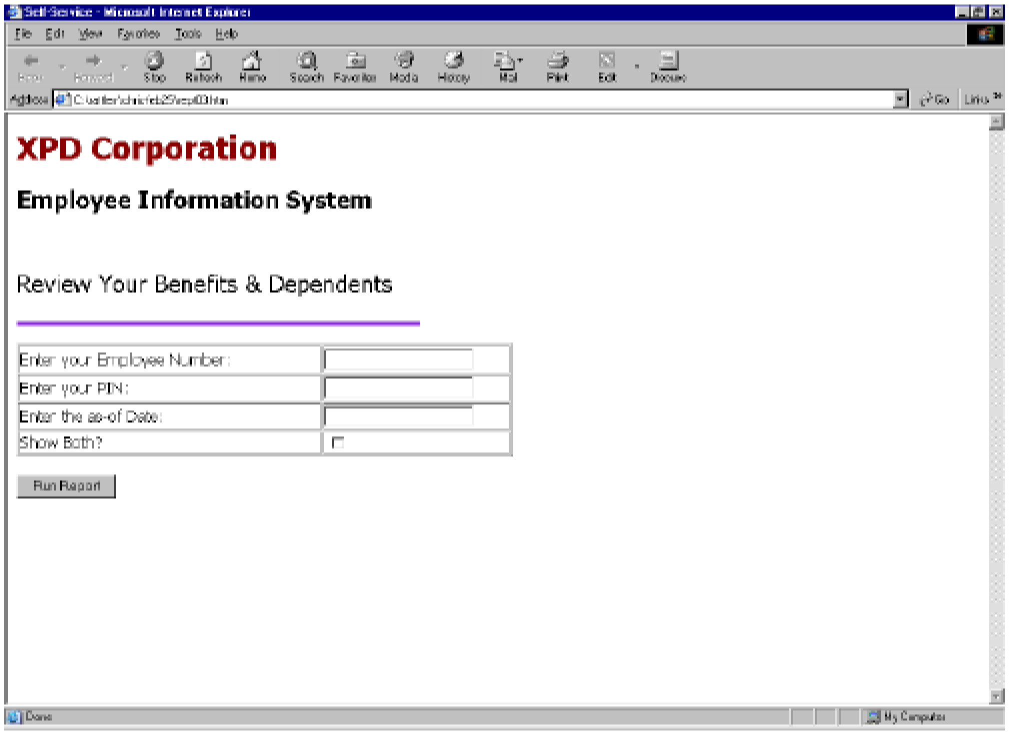
RSSample.ASP

```
<%
if Request("generatereport") = "true" then
     Response.Buffer = TRUE ' For IIS 4 compatibility
     set rr = Server.CreateObject("RSATL.RRWControl")
     rr.reportName = "You must enter a valid report file name.RSW"
     rr.runReport
else
%>

<HTML><HEAD>
<TITLE>Report Viewer Control Document</TITLE>
</HEAD><BODY>

<object WIDTH="85%" HEIGHT="85%"
           CLASSID="CLSID:66960E23-DE25-11CF-876F-444553540000"
            codebase="rrprview.cab#Version=2,0,0,4" id=RepView1>
           <param NAME="LanguageID" VALUE="0409">
           <param NAME="ReportURL" VALUE="<%=
Request.ServerVariables("SCRIPT_NAME") %>?generatereport=true">

           <embed WIDTH="95%" HEIGHT="95%"
           CLASSID="CLSID:66960E23-DE25-11CF-876F-444553540000"
           CODEBASE="rrprview.cab#Version=2,0,0,4"
           TYPE="application/oleobject"
           PARAM_ReportURL=""></object>
</BODY>
</HTML>

<%
end if
%>
```

RattleRR Application Example

A typical RattleRR application might be one where user input is gathered using an HTML page and from that page is then passed to the RattleRR ASP.

Here is an example of an HTML file for an Employee Information application. On this screen the user fills in 4 fields and then clicks a button to run the report.


Self-Service - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address C:\rattler\chris\feb25\rep03.htm
Go Links

XPD Corporation

Employee Information System

Review Your Benefits & Dependents

Enter your Employee Number:	
Enter your PIN:	
Enter the as-of Date:	
Show Both?	☐

Run Report
Done
My Computer

Within this HTML there is form tag:

<FORM action="BenDep.asp" method="get" name="eis">

that points to the BenDep.ASP file that will produce the report. The user input from the HTML is stored to four variables using INPUT tags:

employee number (name=emp)

userid (name=uid)

date for age calculations (name=aso)

checkbox parameter used for line logic within the report (name=shb)

These are then evaluated as report parameter values within the ASP.

Not explicitly specified by user input but sent by a hidden value is the name of the report (name=rep) set within the HTML by the line:

<INPUT TYPE="HIDDEN" NAME="REP" VALUE=BenDepVerifs.RRW">

Within the BenDep.ASP, each variable must be handled first by the “else” section of the ASP (since by default the generatereport parameter is not initially true). The section of the “else” clause below the

 <param NAME=”ReportURL”

statement is where this is defined. There must be one assignment for each parameter passed into the report from the calling web page.

Once defined here, these parameters can be accessed within the ASP script in the initial section of the page by issuing the various Request(“varname”) statements and passing them into variables within the script. Note that some of the values may need secondary processing, either to parse them or put them into a format suitable for reporting.

Contents of BenDep.ASP

```
<%
if Request("generatereport") = "true" then
     Response.Buffer = TRUE ' For IIS 4 compatibility
     set rr = Server.CreateObject("RRATL.RRWControl")

     rr.filterUsage=2

     cemp = Request("emp")
     cuid = Request("uid")
     crep = Request("rep")
     caso = Request("aso")
     cshb = Request("shb")

     rr.parameters("EmpNo") = cemp
     rr.parameters("PINo") = cuid
     rr.parameters("BenAsOf") = caso
     rr.parameters("ShowBoth") = cshb

     rr.filter = "hrp->p_empno = '" & cemp & "'"
     reppath = "C:\InetPub\wwwroot\rr-rept\rr-reports\" & crep
     rr.reportName = reppath
     rr.runReport
else
%>

<HTML><HEAD>
<TITLE>Report Viewer Control Document</TITLE>
</HEAD><BODY>

<object WIDTH="85%" HEIGHT="85%"
     CLASSID="CLSID:66960E23-DE25-11CF-876F-444553540000"
      codebase="../rr-components/rrprview.cab#Version=2,0,0,4"
id=RepView1>
     <param NAME="LanguageID" VALUE="0409">
     <param NAME="ReportURL"
           VALUE="<%= Request.ServerVariables("SCRIPT_NAME") %>
           ?generatereport=true
           &emp=<%= Request.QueryString("emp") %>
           &uid=<%= Request.QueryString("uid") %>
           &aso=<%= Request.QueryString("aso") %>
           &shb=<%= Request.QueryString("shb") %>
           &rep=<%= Request.QueryString("rep") %>">
<embed WIDTH="95%" HEIGHT="95%"
     CLASSID="CLSID:66960E23-DE25-11CF-876F-444553540000"
     CODEBASE="../rr-components/rrprview.cab#Version=2,0,0,4"
     TYPE="application/oleobject"
     PARAM_ReportURL="">
</object>
</BODY>
</HTML>

<%
end if
%>
```

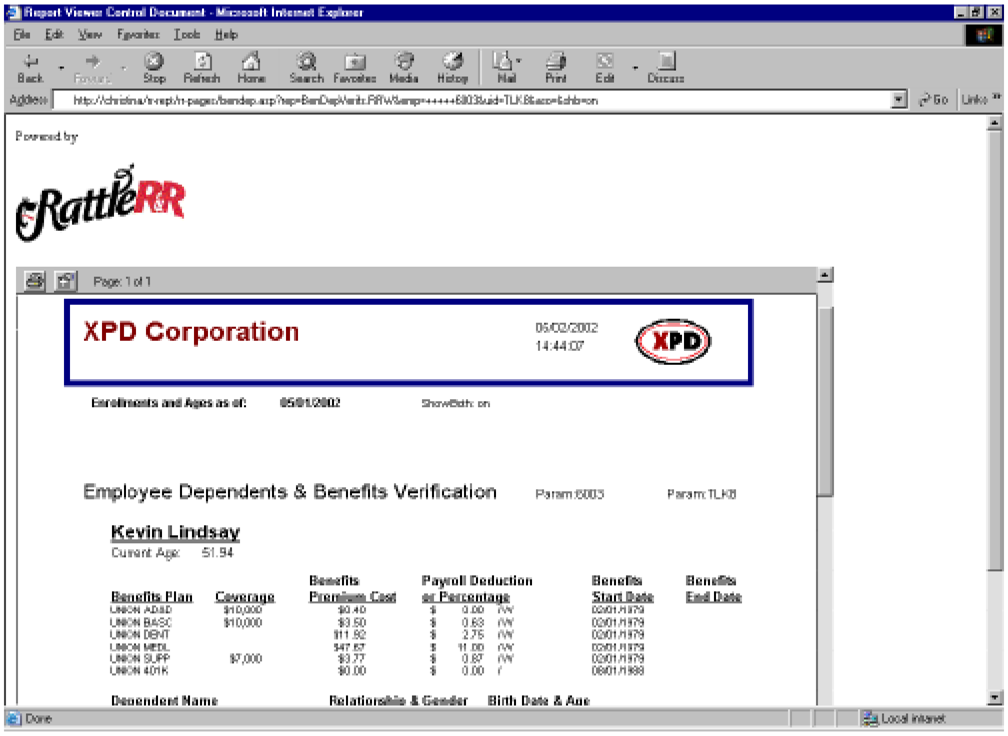
The sequence of events is then that the browser opens the HTML page. The user inputs field values and presses the Run Report button. This launches the RattleRR page BenDep.ASP which in turn runs the report:

C:\InetPub\wwwroot\rr-rept\rr-reports\ BenDepVerifs.RRW

Within this report there are calculated fields that include the expressions:

```
RIPARAM("EmpNo")
RIPARAM("PINo")
RIPARAM("BenAsOf")
RIPARAM("ShowBoth")
```

These calculations will inherit the values that were input on the HTML screen so the report will reflect the choices that were made by the browser client. So when Kevin Lindsay runs the report with an as of date of 05/01/2002 he sees the following report:



Report Viewer Control Document - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites Media History Mail Print Edit Discuss
Address
Go Links
Powered by
Page: 1 of 1
Done
Local intranet

http://christina/rr-rept/rr-pages/bendep.asp?rep=BenDepVerifs.RRW&emp=+++++6003&uid=TLK8&aco=&chb=on

XPD Corporation   06/02/2002 14:44:07

Enrollments and Ages as of: 05/01/2002   ShowBoth: on

Employee Dependents & Benefits Verification   Param:6003   Param:TLK8

Kevin Lindsay
Current Age: 51.94

Benefits Plan	Coverage	Benefits Premium Cost	Payroll Deduction or Percentage			Benefits Start Date	Benefits End Date
UNION AD&D	$10,000	$0.40	$	0.00	/W	02/01/1979	
UNION BASC	$10,000	$3.50	$	0.63	/W	02/01/1979	
UNION DENT		$11.92	$	2.75	/W	02/01/1979	
UNION MEDL		$47.67	$	11.00	/W	02/01/1979	
UNION SUPP	$7,000	$3.77	$	0.87	/W	02/01/1979	
UNION 401K		$0.00	$	0.00	/	08/01/1988	

Dependent Name   Relationship & Gender   Birth Date & Age

RattleRR Properties and Methods

The following topics present detailed descriptions of the properties and methods provided by RattleRR. Each contains the data type returned, a brief description of its functionality and syntax and a usage example.

Common Properties/Methods

- reportName
- filterUsage
- filter
- parameters
- masterTableName
- memoName
- setGroupFieldAndNumber
- setGroupFieldAndNumber

Xbase only

- dataDirectory
- indexExtension
- lowScope
- highScope
- writeAllow
- SetMasterIndex
- setRelationInformation

SQL only

- dataSource
- username
- password

reportName Xbase and SQL

reportName = String

Specifies the report to be executed.

Required element.

You can only use reports stored as single compound files. Reports stored in report library files cannot be used.

Example:

```
rr.reportName = "C:\InetPub\wwwroot\rr-rept\rr-reports\Customer.rrw"
```

Runs the report Customer.RRW

filterUsage Xbase and SQL

filterUsage = Integer

Specifies which report records will be returned.

Available values are:

- 0 - Saved
- 1 - Entire
- 2 - Override

If not explicitly set, the default value is 1 (Entire)

Example:

```
rr.filterUsage = 1
```

Runs report ignoring any saved filter.

filter Xbase and SQL

filter = String

Specifies the record selection criteria for the report.

You must use R&R expression syntax.

The filter expression is only evaluated when filterUsage is set to 2 Override.

Maximum size is 1024 characters.

Example:

```
rr.filterUsage = 2
rr.filter = "hrp->p_empno = '30362'"
```

Selects records where the character field hrp->empno is equal to 30362.

This filter is used in place of any filter that has been saved with the report.

parameters Xbase and SQL

parameters(index as String) = newVal as String;

Used to supply values for RIPARAM() field expressions. Use one line for each parameter value.

If you specify a parameter that does not appear in the report, you will get an “Undefined job control variable” error in the browser’s report window.

Example:

```
rr.parameters("P1")="Hello from P1"
rr.parameters("P2")="Hello from P2"
```

Calculated expression RIPARAM(“P1”) will evaluate to “Hello from P1”

Calculated expression RIPARAM(“P2”) will evaluate to “Hello from P2”

masterTableName Xbase and SQL

masterTableName = String

Used to override the master file that is saved with the report.

Example:

rr.masterTableName="C:\myfiles\customer.dbf"

Uses C:\myfiles\customer.dbf in place of the saved master file.

memoName Xbase and SQL

memoName = String

Used to override any text file that is saved with the report.

Example:

rr.memoName="C:\myfiles\newdata.txt"

Uses C:\myfiles\newdata.txt in place of the saved text file.

setSortFieldAndNumber	Xbase and SQL
setGroupFieldAndNumber	Xbase and SQL

setSortFieldAndNumber fieldName as String, sortNumber as Integer

setGroupFieldAndNumber fieldName as String, groupNumber as Integer

Used to supply sort and group fields to report.

If you specify invalid field name or group number report, you will get an “Unknown or ambiguous sort/group field” or “Invalid sort/group field number” error in the browser’s report window.

Examples:

```
rr.setSortFieldAndNumber "Company","1"
rr.setSortFieldAndNumber "LAST_NAME","2"
rr.setGroupFieldAndNumber "LAST_NAME","1"
```

Adds Company as sort field1, LAST_NAME as sort field 2 and LAST_NAME as group field 1.

dataDirectory Xbase only

dataDirectory = String

Use dataDirectory to replace the default data directory specified in RRW.INI. Note that this directory is only used to locate files when they cannot be found using the saved file path. It does not override the saved file path.

Example:

```
rr.dataDirectory="C:\data\file2002"
```

Search for data files in C:\data\file2002" when they cannot be found in the saved location.

indexExtension Xbase only

indexExtension = Integer

The index extension is used to locate indexes that are specified without extensions or that cannot be found using the extensions saved with the report.

The possible values and meanings are:

0	none
1	CDX
2	IDX
3	MDX
4	NDX
5	NSX
6	NTX
7	WDX

Example:

```
rr.indexExtension=1
```

If a saved index has file name CUST.MDX and this index cannot be found, use CUST.CDX instead.

scopeUsage Xbase only

scopeUsage = Integer

Used to control starting/ending master file records based on master index key.

Available values are:

0 - Saved

1 - Entire

2 - Override

If not explicitly set, the default value is 1 (Entire)

Example:

```
rr.scopeUsage=0
```

Use any scope start/end values that are saved with the report.

lowScope Xbase only

lowScope = String

Used to control starting value for master file records based on master index key.

The lowScope value is only evaluated when scopeUsage is set to 2 (Override).

Example:

```
rr.scopeUsage=2
rr.lowScope ="A"
```

Start reading master file records beginning with value A.

highScope Xbase only

highScope = String

Used to control starting value for master file records based on master index key.

The highScope value is only evaluated when scopeUsage is set to 2 Override.

Example:

```
rr.scopeUsage=2
rr.highScope ="D"
```

Stop reading master file records after value D.

writeAllow Xbase only

writeAllow = Boolean

Used to control shared access to database files.

Available values are:

0	Report cannot access files in write use by other apps
non-zero	Report will access files in write use by other apps

Example:

```
rr.writeAllow=1
```

Allow database users to make updates to data files while report is in use.

setMasterIndex Xbase only

setMasterIndexInformation indexName as String, indexType as String, tag as String

Used to supply a master index to a report. The index type may be N (numeric), D (date) or C (character).

Example:

```
rr.setMasterIndexInformation" C:\myfiles\customer.cdx ",”N”,"CUSTNO"
```

Sets master index to the numeric tag CUSTNO of the C:\myfiles\customer.cdx compound index file.

setRelationInformation Xbase only

setRelationInformation filepath as String, indexpath as String, tag as String, alias as String, aliasNumber as Integer

Used to change a related file in the report.

Examples:

```
rr.setRelationInformation
"C:\rrw9\sample\ord2002.dbf”,”C:\rrw9\sample\ord2002.cdx”,"custno",”ord2001”,”1”
rr.setRelationInformation
"C:\rrw9\sample\inv2002.dbf”,”C:\rrw9\sample\inv2002.cdx”,"custno",”inv2001”,”2”
```

Changes related file for current related alias ord2001 to ord2002.dbf using tag custno of ord2002.cdx and changes related file for current related alias inv2001 to inv2002.dbf using tag custno of inv2002.cdx

datasource SQL only

dataSource = source as String

Use to specify the ODBC data source parameter for the report.
The report data source must be a System data source and NOT a user or file data source since the browser client does not come in to the server as an authenticated user.

Example:

```
rr.dataSource="Oracle7 Tables"
```

Run report using the Oracle7 Tables datasource.

username SQL only

username = username as String

Use to specify the username to be used in connecting to the data source associated with the report.

Example:

```
rr.username="Scott"
```

Log in to data source using user Scott.

password SQL only

password = password as String

Use to specify the password to be used in connecting to the data source associated with the report.

Example:

```
rr.password="tiger"
```

Log in to data source using password tiger.

Troubleshooting and Known Issues

Troubleshooting problems

A prerequisite to successfully using RattleRR is to ensure that the Data Warehouse foundation is in place and that reports can be successfully run using the runtime executable. You should refer to the appropriate Data Warehouse documentation for instructions on configuring and creating reports.

When creating reports to be run via RattleRR where you want to allow for parameter prompting, you need to use RIPARAM() based calculated field instead of parameteRR fields since parameteRR are designed to return dialog to the client screen which for RattleRR is actually the web server and not the local browser.

It is also important the server must have a "clear line of sight" to the data being used in the report. This is important for file-based reports such as those Xbase version and any SQL data sources that point to specific database locations on the network. Using standard PC path notation and verification for your reports works fine as long as everything resides on the server. However, for data that is remote (i.e. mapped drives), it cannot be guaranteed that the anonymous internet user id will be able to "see" that data as well. However, data that is visible over the network, but stored on another server should be accessible via the UNC path back to the data (\\servername\sharename\pathtodata\datafile.ext).

RattleRR uses the RRPRVIEW.CAB ActiveX component to deliver reports. You can test the functionality of this component by using either the Report Designer or runtime to create and export the contents of a report to an ActiveX PDI file along with an HTML container file with pointers to that PDI and to the RRPRVIEW.CAB. You should refer to the appropriate Data Warehouse documentation for instructions on ActiveX export. You can then access the HTML from a browser to verify that the CAB can be correctly downloaded and that the static PDI file can be viewed.

When a RattleRR ASP page is launched from a browser, if the local machine does not have the ActiveX viewer control installed, then the user will be prompted to download and install the control. Once the control is locally available, the report page will display an empty report window as the report is generated to a temporary PDI file on the server. When the file is complete, the report output will appear in the report window.

There are however a number of things that may go wrong in this process. Some are as simple as a minor typographical error in the ASP page and others may require more extensive investigation.

Some problems will result in an error message display within the browser report window. For example an improperly set TEMP environment variable or an invalid report name will be reported to the browser. In general these errors should be reasonably easy to resolve based on the specific error. Other error conditions may instead be reported at the server so you need check both client and server if any problems result. Other error conditions may never return an explicit error and the browser window may simply fail to display a report.

Troubleshooting Checklist

The following troubleshooting checklist should help you to find and correct errors.

- Make sure that there is only one copy of R?WRUN.EXE and R?RRPT32.DLL on your server and that the installed version of these files come from the RattleRR installation. Having earlier versions of these files can cause a variety of unexpected conditions and can prevent successful DLL registration of the RattleRR components.

- Verify that the R?WATL.DLL files have been correctly installed and registered. You can manually register these files by changing to the directory where they reside and running:

  ```
  REGSVR32 RRWATL.DLL
  REGSVR32 RSWATL.DLL
  ```

  for the Xbase and SQL versions respectively.

- Verify that the report can run via conventional runtime on the server.

- Verify that the ActiveX component can be successfully downloaded from the server.

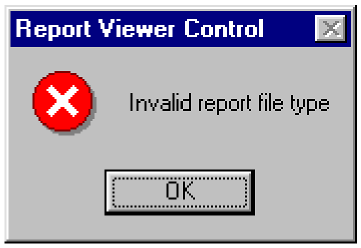
- If the problem occurs across all reports, start with a simple single file reports as the smallest test case.

- Setting a property to an invalid value can result in an "invalid report control object" error.

- For SQL version, make sure that you are using a System data source and not a User data source.

- If a report is saved with the File->Print Print to file box checked the report may either not display in the browser or the browser may get the invalid report file type error show below.

  Note that this error can be returned under other conditions as well so troubleshooting may take some careful investigation of the problematic report.


Report Viewer Control

Invalid report file type

Known Issues

- The Xbase setMasterIndexinformation method is currently non-functional.
- Setting filterUsage=0 in the Xbase version returns all records rather than those selected by the saved query.
- Images only display when they are stored in the ASP folder.

Technical Support

Technical Support service gives you access to product experts who can help solve problems you encounter using R&R Report Writer products.

Several options provide you with the level of service you need to get answers in the most timely and cost-effective manner.

Capabilities

Liveware Publishing offers varying levels of service to meet your technical, administrative, and budgetary requirements. From comprehensive support plans to self-service online resources, we've got the right program to fit your needs.

Key Benefits

Help available when you need it — online, email or 800#.

Dedicated, knowledgeable staff — over 50+ years R&R experience.

Cost-effective — pay only for what you use.

Option 1 —Pay-per-Call

At any time between the hours of 8:30 AM and 5:30 PM Eastern Time, you may call 302-791-9446 to speak with a technical support representative.

The fee for support is $100 per hour, with a minimum charge of $10 per call.

Payment can be made by credit card (VISA/MC/AMEX). You will not be charged for problems arising from R&R Report Writer software.

(Time is calculated to the closest five dollar time unit, only for time actually spent by a Liveware Publishing support representative resolving the issue. Any time spent is subject to Liveware Pu blishing management review for effectiveness of the support provided.)(A "call" is defined as phone conversations, e-mail, and voice-mail messages related to resolution (or attempted resolution) of a particular issue. Issues may be grouped in a particular "call" to reach the minimum.)

Option 2 —Support/UpgradeAccount

You can establish a pre-paid support/upgrade account (SUA) with Liveware Publishing to receive a faster response and at a discounted rate.

The cost for opening an SUA account for a single user is $125, and special pricing is available for multiple user accounts.

Once you have opened an SUA account, at any time between the hours of 8:30 AM and 5:30 PM Eastern Time, you may call the special 800 number to speak with a technical support representative. You may also send an email if you are working outside our normal business hours.

The fee for support is discounted to $80 per hour, with a minimum charge of $10 per call. You will not be billed for problems arising from R&R Report Writer software.

In addition, with the purchase of an SUA account you receive the following benefits:

- Installation support during the first 60 days is not chargeable.
- One FREE tech support call.
- One FREE copy of Relate and Report: Your Guide to Reporting with R&R. (350 pages, a $59 value.)
- FREE maintenance releases of R&R Report Writer software.
- Reduced pricing on product upgrades. You may apply any unused portion of your SUA account balance to the purchase of upgrades.
- The balance in your account never expires.

Customers with SUA accounts will be supplied a statement of account usage, upon request, itemizing time spent on each call. Per-call customers will be informed of the time and cost of the call at its conclusion. Requests for review of charges may be sent via e-mail at our web site. Customers with SUAs should deposit sufficient funds to cover reasonable support needs for your organization for at least three months. Automatic re-billing to replenish accounts is available, or you may use a credit card or purchase order.

Option 3 —Self Service Resources

At any time, you may refer to the Resources page on our web site http://www.livewarepub.com for Documentation, Frequently Asked Questions, Technical Bulletins, Service Packs, and other information about R&R Report Writer.

The site also includes an R&R User Discussion Forum with a section to specifically discuss RattleRR so that you can post questions and view posting from other users.

SubscribeNow!

Complete the form on our web site http://www.livewarepub.com/ts_order.htm to open your Support/Upgrade Account today, or call 800-936-6202 or email us at livesales@livewarepub.com for more information.