FIXING WHAT SHOULD WORK, BUT DOESN'T

THE RDL COOKBOOK

FREE EDITION

Get the complete fix for the most-searched SSRS failure, and see the 34 others the full edition solves.

(The full edition sold separately)

INSIDE

- Contains a list of all 35 problems the full edition solves
- See the actual server errors. Is one of them yours?
- A complete fix for a common failure usually solved incorrectly
- The internet's top fix, and why it still prints the wrong page
- Learn how all claims are proven before they're included

SAMPLE RECIPE · REPRODUCED ON A LIVE SSRS 2022 SERVER

PROBLEM	Globals!PageNumber in a report body textbox
ISSUE	rsPageNumberInBody These global variables can be used only in the page header and page footer.
SOLUTION	<PageFooter>: a child of Page, before PageHeight

1 complete chapter · 35 problems listed · Verified on SSRS 2022

BY Rob Joseph

RobCoTek

THE RDL COOKBOOK · FREE EDITION

Do any of these look familiar?

35 report failures the book covers, and one of them solved in full

> On the next page is every problem the book takes on, written the way it sounds when you are the one living through it. Read the list. If three or four of them are yours, you already know whether this book is for you, and you did not have to take anybody's word for it.
>
> After the list is one chapter exactly as it ships: the most-searched SSRS complaint there is, in full, including proof that the workaround the internet gives you is broken. Nothing is summarised. In front of it is the method section, which is what separates this from the answer you already found and could not get to work.

RobCoTek

What the book covers

Every line below is a chapter. They are written the way the problem sounds when you are living it, not the way the chapter is titled, because at four in the afternoon nobody searches for "aggregate scope", they search for what is on the screen in front of them. Where the server gives an error, it is printed underneath in red, word for word, so you can recognise your own without reading a line of prose. 35 chapters, 7 parts, and every one of them was deployed and rendered on a live SSRS 2022 server before it made the page.

VALIDITY & STRUCTURE (WHY AN RDL WON'T OPEN OR DEPLOY)

1. Something is wrong and there is nothing on screen that could be wrong. The designer shows a valid report, the server accepts it, and the output still does not match what the definition plainly says.
2. The report opens perfectly on the machine that authored it and the server refuses to take it at all. You did not change anything in the file.
   ```
   "The definition of this report is not valid or supported by this version of Reporting Services. The report definition may have been created with a later version..."
   ```
3. A file the internet insists must fail deploys clean, and a file that looks correctly ordered is rejected anyway. You cannot find out which rule is real, and half the advice you find contradicts the other half.
   ```
   "...has invalid child element '...'. List of possible elements expected: ..."
   ```
4. Every example you can find is a fragment, a screenshot, or a file that will not deploy. There is nowhere to start from something known to work.
5. The server rejects the file and points at the body, or at a margin, or at a number, and every element it names looks perfectly correct on its own.
   ```
   "0 does not contain a unit specification."
   ```
6. The designer reports an error on a line number that no longer exists in the file, and you have now spent an hour editing three lines that were never the problem.

DATA SOURCES & DATASETS

7. The server has to change per environment or per parameter, and the shared data source will not accept an expression. Every version you try either refuses to save or connects to the wrong box.
8. The report totals correctly at one level and is wrong at every level above it, and adding a drill-down means writing a second query. Separately, it pulls every row across the wire and then discards most of them.

PARAMETERS

9. The multi-select works against a plain query and returns nothing the moment the dataset calls a stored procedure. No error, no warning, just an empty result set.
10. The parameters are right, the datasets are right, and the server will not accept the report at all. It never gets as far as rendering, so nothing you can preview tells you anything.
    ```
    "The report parameter 'City' has a DefaultValue or a ValidValue that depends on the report parameter 'Region'. Forward dependencies are not valid."
    ```

11. The report ran fine last month and today it will not run itself. Nothing in the definition changed. It opens to the parameter bar and sits there, and the render URL answers with a 500 and an empty body.

    `"This report requires a default or user-defined value for the report parameter 'City'." (and an HTTP 500 with no body from the render endpoint)`

EXPRESSIONS & CALCULATIONS (MARQUEE; GUI CAN'T HELP HERE)

12. A percent-of-total, or an average of subtotals. The designer lets you type it and the server will not process it.

    `"...uses an aggregate function on a report item. Aggregate functions can be used only on report data, not on report items." / "...has a nested aggregate that specifies a dataset scope. Inner aggregates cannot specify a dataset scope."`

13. A running total that restarts somewhere you did not ask it to, or that accumulates in an order different from the one on screen.
14. The period-over-period column is correct on every row except the first, which shows the whole value where you expected a blank. It reads as a data problem and it is not.
15. Two datasets you cannot join in SQL, one value you need from the other, and no join anywhere in RDL. When you do get a value back it is only ever the first match, and totalling it is rejected outright.
16. The grand-total row shows #Error or the wrong label while the group rows above it are fine, and the count is larger than the number of things you were counting.

    `"#Error" (in the total cell, with no other failure reported)`

17. A matrix where every column should shade against its own range, and each nested aggregate you try is either rejected or silently scoped to the wrong thing.

LAYOUT, FORMATTING & RENDERING (THE "SHOULD WORK BUT DOESN'T" CORE)

18. You put the page number in a body textbox and the server will not deploy the report. The workaround the whole internet links to deploys clean and prints 0 on every page.

    `"The Value expression for the textrun '...' refers to the global variable PageNumber or TotalPages. These global variables can be used only in the page header and page footer." (surfaced by some tools as rsPageNumberInBody)`

19. It previews fine, and every page of the export or the printout is followed by a blank one. A four-page report comes out at eight. Nothing errors, and going back to the designer shows nothing wrong.
20. The header row appears on page one and nowhere else, with RepeatOnNewPage set exactly as documented, accepted at deploy, and no error or warning anywhere.
21. Alternating row shading and red negatives are one click in Excel, and in RDL there is no row to put them on and no stylesheet to hang them from.
22. The toggle either does nothing, or it hides rows and gives you no way to open them all at once. Then someone exports the collapsed report and every hidden row is in the file.
23. Currency symbols and date formats that are correct on your machine and wrong on the server, or correct until the report is opened somewhere else in the world.
24. A minimal hand-authored chart, rejected over and over, each attempt naming a different element you have never heard of.

    `"The report definition element 'ChartMember' is empty ... It is missing a mandatory child element of type 'Label'." / "The element 'ChartCategoryAxes' ... has invalid child element 'ChartCategoryAxis' ... expected: 'ChartAxis'." / "true is not a valid value."`

25. The logo appears in preview and comes out as a broken red X on the server. Nothing failed, nothing errored, the rest of the report is perfect. Sometimes it works for one user and not another.

NAVIGATION, NESTING & EXPORT

26. The links work in preview and are dead in the exported PDF. Nobody can tell you which of the three kinds of link you actually built.
27. The nested layout is exactly right, and on real data the report takes minutes. Nothing in it looks expensive, and the designer shows a tidy little box.
28. Export to Excel and get one sheet where you wanted one per group, or the right number of sheets all named Sheet1, Sheet2, Sheet3.
29. The PDF comes out at the wrong paper size, or in a font nobody chose, and it was right in preview on the machine you authored it on.
30. The CSV a downstream system has to consume arrives with columns called Textbox3 and Textbox7, and the title and the logo are in there as columns too.

TARGETS, GENERATION & THE FUTURE (DURABLE DIFFERENTIATORS)

31. A .rdlc the report server will not accept, or a .rdl loaded into a ReportViewer that renders an empty region. It is the same XML either way and neither one runs in the other's home.
    ```
    "The data source '...' has both or neither of the following: DataSourceReference and ConnectionProperties. DataSource must have exactly one of these elements." / "An error occurred during local report processing..."
    ```
32. Code that emits a report which looks right in every respect and is rejected by the server anyway, usually over a tablix whose pieces do not agree with each other. Every fix moves the rejection somewhere else.
33. Columns you cannot know until the query runs, and a hand-built matrix that renders only its first column or is rejected outright.
    ```
    "the tablix has an invalid structure"
    ```
34. You asked an AI for an SSRS report and got a file that will not open. Or worse, one that opens, deploys clean, and is quietly wrong in a way nobody notices for a month.
35. A report that ran for years on SSRS will not publish to the paginated service, and the service answers with the same unhelpful "not valid or supported" string you never saw on-prem.
    ```
    "rsInvalidReportDefinition" / "The definition of this report is not valid or supported..."
    ```

Three or four of these familiar? That is the book. What follows is one of them, answered in full, so you can judge the other thirty-four by it.

How this book handles "it should work but it doesn't"

Most of the pain in RDL is not a typo. It is doing the obvious, reasonable thing and having it silently fail. You checked the box, and the headers still don't repeat. You set the property, and the page number in the body errors. The internet is full of confident answers to these, and many are simply wrong (the

most-copied page-number-in-body "fix" returns 1 on every page). So this book treats the whole category with one method.

The root cause is always the same shape: the designer (or the property name) is an abstraction, and the obvious action operates on the *wrong hidden layer*. Fix the layer, at the XML level where it is visible, and prove it renders. When correct-looking RDL misbehaves, the cause is almost always one of seven hidden layers. Check them in this order:

1. Render order: a render-time fact used in a data-time context (`Globals!TotalPages` in the body).
2. Scope: the expression evaluates in the wrong grouping scope (wrong totals; nested-aggregate rules).
3. Which element carries the property: a GUI toggle mapped to the wrong/insufficient element (RepeatColumnHeaders does nothing; the static `TablixMember` must carry it).
4. Geometry arithmetic: `Body.Width <= PageWidth - margins` ; trailing whitespace → blank pages.
5. Data-dependent silent failure: a data value breaks valid config (a NULL kills "Select All").
6. Environment / identity: preview runs as you; the server runs as a service account (external image blank on the server).
7. Missing feature: the built-in you expect doesn't exist (expand-all/collapse-all needs a pattern).

Every "should work but doesn't" recipe in this book follows the same structure: what you tried → the exact symptom → which hidden layer → the fix in the XML → proof it renders (and a reproduction of the failure). And when SSRS genuinely can't do the thing, this book says so and gives the least-bad pattern with its limits, because a verified "here's why it's impossible and the closest you can get" beats one more broken workaround.

"Page X of Y" and why you can't put the page number in the body

The flagship "it should work but it doesn't" chapter. Everything below was reproduced on a live SSRS 2022 server. The screenshots are real renders, not mock-ups. Samples: `samples/05_pagenum_body_fail.rdl` , `samples/06_pagenum_footer_works.rdl` , `samples/07_pagenum_body_hack_broken.rdl` .

What you tried

You have a report and you want it to say "Page 3 of 12", or you want the current page number to appear next to a row in the body (common for invoices: "Invoice continues, page 2"). So you put this in a body textbox:

```
="This report has " & Globals!TotalPages & " pages"
```

Reasonable. `Globals!PageNumber` and `Globals!TotalPages` exist. You referenced them like any other global. It does not work.

The symptom

The server rejects the report at deploy time (you don't even get to render):

> The Value expression for the textrun 'Textbox1...' refers to the global variable PageNumber or TotalPages. These global variables can be used only in the page header and page footer.

(Some tools surface this as the `rsPageNumberInBody` error.) This is a hard rule, not a glitch.

Why it fails: hidden layer #1: render order

SSRS processes a report in two passes: a data/layout pass (the body is built from the dataset), then a render/pagination pass (the engine flows that body onto physical pages and only *then* knows page numbers and the total). `PageNumber` and `TotalPages` are facts that do not exist yet when the body is evaluated. The body is built before pagination happens, so the page number is literally unknowable there. That is why the engine forbids it outright rather than returning a wrong value. Page headers and footers are the exception: they are evaluated per physical page during the render pass, when the numbers finally exist.

The fix: put it in the page footer (this works)

Move the expression into a `PageFooter` (or `PageHeader`). In hand-authored RDL the footer lives on the `Page` element, and `PageHeader` / `PageFooter` come before `PageHeight` in the child order:

```
<Page>
  <PageFooter>
    <Height>0.4in</Height>
    <PrintOnFirstPage>true</PrintOnFirstPage>
    <PrintOnLastPage>true</PrintOnLastPage>
    <ReportItems>
      <Textbox Name="FooterPage">
        <Paragraphs><Paragraph><TextRuns><TextRun>
          <Value>="Page " &amp; Globals!PageNumber &amp; " of " &amp; Globals!TotalPages</Value>
        </TextRun></TextRuns></Paragraph></Paragraphs>
        <Top>0.1in</Top><Left>0in</Left><Height>0.25in</Height><Width>6.5in</Width>
      </Textbox>
    </ReportItems>
  </PageFooter>
  <PageHeight>11in</PageHeight>
  <PageWidth>8.5in</PageWidth>
  <LeftMargin>1in</LeftMargin><RightMargin>1in</RightMargin>
  <TopMargin>1in</TopMargin><BottomMargin>1in</BottomMargin>
</Page>
```

Proof (real render of a 3-page report):

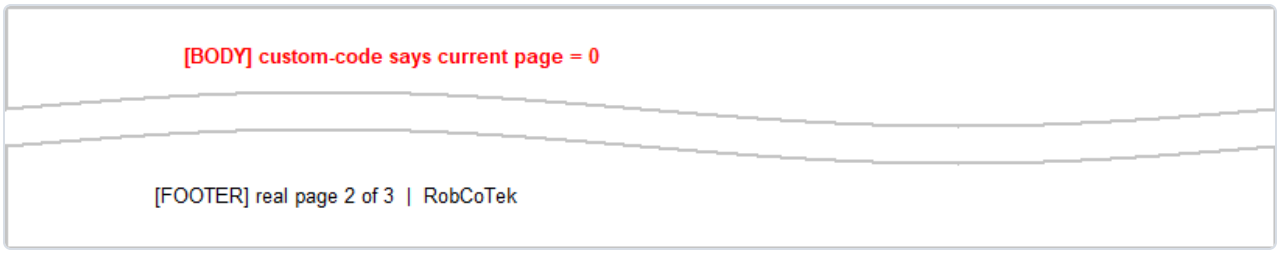
The footer correctly reads "Page 1 of 3", then "Page 2 of 3" on the next page. Verified.

The broken workaround you'll find online, and proof it's broken

The most-copied "get the page number into the body anyway" trick uses custom `Code`: a footer textbox calls a `SetCurrentPage(Globals!PageNumber)` function, and a body textbox calls `GetCurrentPage()`:

```
Public CurrentPage As Integer = 0
Public Function SetCurrentPage(ByVal p As Integer) As String
    CurrentPage = p
    Return ""
End Function
Public Function GetCurrentPage() As Integer
    Return CurrentPage
End Function
```

It deploys clean (no error), which is why people believe it works. It does not. Here is physical page 2 of the report: the footer (evaluated during render) correctly says "real page 2 of 3", while the body (calling the exact same shared variable) says current page = 0:



Same hidden layer #1: the body is evaluated before the footer runs, so `GetCurrentPage()` returns the initial `0` (on other reports it returns a stale value from a prior page, equally wrong). The custom-code trick cannot beat render order. It just moves the wrongness somewhere the deployer won't

immediately notice.

The honest bottom line

The current page number genuinely cannot be shown in the report body. It is not a limitation you can code around. The information does not exist at the moment the body is built. Anyone showing you a body-based page counter is showing you a bug that hasn't bitten them yet.

If you truly need per-page text in the body region (invoices, "continued on next page"), the correct approaches are:

- Put page numbering in the header/footer, where it belongs. (Above.)
- For "page N of this record's section," don't use page numbers at all: use group-scoped page breaks and `Globals!OverallPageNumber` / `ResetPageNumber` patterns in the footer (covered in the page-break chapter), which count pages *per group* at render time.

That is the whole truth, verified: one working place (the footer), one hard rule (render order), and one popular workaround that is simply broken.

The rest of the book

35 chapters across 7 parts: validity and structure, data sources and datasets, parameters, expressions and calculations, layout, formatting and rendering, navigation, nesting and export, and targets, generation and the future. Every recipe names the sample that proves it, and every sample was deployed and rendered on a live SSRS 2022 server before the recipe was allowed to make a claim.

Negative samples are the point. Where the widely repeated fix does not work, the recipe says so and ships the broken file so you can watch it not work, then gives the one that does. The companion Tools + Asset Pack carries 46 sample files: 35 that deploy and render, 10 broken on purpose, and one `.rdlc`, alongside `rdl_validate.py`, an offline linter that catches the structural traps before you deploy, and `gen_rdl.py`, a generator built around a self-validation loop.

The book is not on sale yet. It is finished and in the hands of a licensing office, which takes as long as it takes. Put an email address on the pre-release list and you will hear once, on the day it opens, with a discount code that belongs to you and works one time. No other mail, and one reply to leave.

Sign up at robcotek.com/rdl