



Thousands of Tables and Four Farthings

Hacking Your Way to a Better Report

The Great ILS-Data Pre-Conference

Daniel Messer
Polaris ILS Administrator
Library Systems & Services

Hello, I'm Dan

- I like bright text on dark backgrounds



Hello, I'm Dan

- I like bright text on dark backgrounds
- 30 years in libraries
- Polaris ILS admin for Library Systems & Services
- SQL Hacker (as will soon be made clear)





<https://cyberpunklibrarian.com/iug2026>



Beginning at the Beginning – WTF is Four Farthings?

- De-centralized central data server
 - Wait what?
- I have a weird relationship with libraries and Polaris
- I do not help manage Polaris for a library or a consortium
- I help manage Polaris for 27 libraries
 - 10 of which are in consortia (of which there are two)
- We would like to be able to do things with the data between them
- But we don't want to move that data around



Okay, sure but why Four Farthings?

- My boss is a huge fan of *Lord of the Rings*
- Me, I love *Lord of the Rings*
- But she has forgotten more about *Lord of the Rings* than I know
- Rumour has it that she read *The Silmarillion* and enjoyed it
- All that to say, is that The Shire is divided into four parts
- The Four Farthings



The Problem

- The Finance Department
- No that's it, they're the problem
- Okay they run this report to get paid invoices, right?
- But do they do it through SSRS?
 - No
- Through the Polaris reporting system?
 - No
- SimplyReports?
 - No



The Problem – Continued

- No, they run a custom SQL query in the Find Tool
- Select all the Invoices
- Right-click
- Select Print -> Full
- A PDF appears from... somewhere
- Yeah...



| Type | Payment M... | Status | Date |
|---------|--------------|--------|----------|
| Regular | Purchase | Paid | 1/2/2026 |
| Regular | | d | 1/2/2026 |
| Regular | | | |
| Regular | | | |
| Regular | | | |
| Regular | | | |
| Regular | | d | 1/2/2026 |
| Regular | Purchase | Paid | 1/2/2026 |
| Regular | Purchase | Paid | 1/2/2026 |
| Regular | Purchase | Paid | 1/2/2026 |

Open

Print >

Links >

Delete

Properties

List View

Invoice Voucher (Summary)

Invoice Voucher (Full)

Analysis

- So this is obviously coming from some kind of internal reporting process
- But where is it and how does it work?



Solution



Where's the Call Coming From?

- Inside the house, obviously
- So let's trace the call



SQL Server Profiler

- A separate programme that works with SQL Server
- Allows you to do a SQL Trace
 - Watch everything that's happening in your database
- Set up and run the search in the Find Tool
- Start the trace
- Select and print the full invoice vouchers
- Stop the trace
- Look for clues

Interesting Finds

```
exec ObjectExists @Path=N'/Polaris/System/Internal/Vouchers1',@AuthType=1
```

```
exec Polaris.Rpt_Vouchers1  
@InvoiceID=1,@PayHistoryID=1,@sTempTable=N'##6_861'
```

Ran a search on that @Path and, oh hey:

C:\ProgramData\Polaris\{POLARIS VERSION}\Reports\System\Internal



SQL Server Extended Events

- The SQL Profiler Trace casts a very wide net but...
- Now we have some idea of which fish we want to catch
- Time for Extended Events
- Set up a search
 - sql_batch_completed
 - rpc_completed
 - These capture sproc calls and ad-hoc queries
 - Filter by username. In this case: polaris



More Interesting Finds

- Looking through the captured data, you can see the order
- declare @p3 nvarchar(255) set @p3=N'##1158_861' exec
Polaris.Rpt_CreateTempTable 861,1158,@p3 output select @p3
- exec Polaris.Rpt_Vouchers1
@InvoiceID=1,@PayHistoryID=1,@sTempTable=N'##1158_861'
- exec Polaris.Rpt_VoucherLines @invoiceid=317773,@payhistoryid=325569



So what did we discover?

- There's an internal report, not shown in SSRS, called Vouchers1
- There are three sprocs involved:
 - Rpt_CreateTempTable
 - Rpt_Vouchers1
 - Rpt_VoucherLines
- And we can look at the way the sprocs are called and compare that with their code to find out what parameters they take
- Rpt_CreateTempTable takes the PolarisUserID, WorkstationID, and a table name
 - Which means it creates a predictably named table



Know the Process, Hack the Process

- We need to create our own temp table

```
EXEC Polaris.Polaris.Rpt_CreateTempTable 861,1158,@p3 OUTPUT;
```

- Populate the table with our own SQL query, the same one Finance was using but with variable start and end dates

```
EXEC Polaris.Polaris.Rpt_Vouchers1 @InvoiceID = 1, @PayHistoryID = 1,  
@sTempTable = N'##1158_861'
```

- But wait! What about Rpt_VoucherLines?
 - Turns out, it's only used as a sub-report in the Vouchers1 RDL file



```

CREATE PROCEDURE [Lss].[Rpt_PrintInvoices]
    @StartDate DATETIME,
    @EndDate DATETIME
AS

BEGIN
    SET NOCOUNT ON;

    -- Set up a global temp table for collecting InvoiceIDs. This table is predictably named based on the
    -- WorkstationID and PolarisUserID. While it's initially set up as below, the RPT_CreateTempTable
    -- sproc will adjust the naming.
    DECLARE @p3 nvarchar(255) SET @p3=N'##1158861';

    -- 861 = PolarisUserID for Dan, 1158 = Dan's WorkstationID, @p3 provides a variable for the temp table
    EXEC Polaris.Polaris.Rpt_CreateTempTable 861,1158,@p3 OUTPUT;

    -- Populate the temp table created by the sproc above
    INSERT INTO ##1158_861

    SELECT
        ph.invoiceID
    FROM
        Polaris.Polaris.PaymentHistories ph
    INNER JOIN
        Polaris.Polaris.Invoices i
        ON (i.InvoiceID = ph.InvoiceID)
    WHERE -- Invoice status of PAID
        i.InvoiceStatusID = 5
    AND
        i.InvStatusDate between @StartDate and @EndDate
    GROUP BY
        ph.InvoiceID HAVING COUNT (ph.invoiceID) = 1;

    -- Pass the temp table to the sproc below to pull invoice related data
    EXEC Polaris.Polaris.Rpt_Vouchers1 @InvoiceID = 1, @PayHistoryID = 1, @sTempTable = N'##1158_861'

    -- Tidy up
    DROP TABLE IF EXISTS ##1158_861;

END

```

Thank you!

