

Denkh HTML Reporter

Web Based Report Generation Software

Written By Scott Auge sauge@amduus.com scott_auge@yahoo.com

Amduus Information Works, Inc.

<http://www.amduus.com>

Table of Contents

<i>License</i>	5
<i>What is it?</i>	6
<i>Basic Software Requirements</i>	8
<i>Basic Report User Requirements</i>	8
<i>Basic Report Designer Requirements</i>	8
<i>Advanced Report Designer Requirements</i>	8
<i>Installing the software</i>	9
<i>Basic Components of Report</i>	13
Page Header	13
Column Headers	13
Record	13
Summary	14
Page Footer	14
Query	14
<i>Basic Running of a Report</i>	15
<i>Creating a report</i>	17
<i>When things go wrong</i>	24
<i>Examples:</i>	26
A more complex report	26
<i>A letter report</i>	32
<i>More advanced subjects</i>	34
Calculations	34
Totaling and sub-totaling	35
Page Breaking	35
Using Hyperlinks in the @plc() macro's for "Lookups."	35
Drilling into other reports	35

Macros	36
@calc(Formula,Format)	36
@date()	36
@drill(NameOfReport,Arg1 Arg2,Target,HyperLinkText)	36
@dur()	37
@emb(ReportName,ReportArg1 ReportArg2 ...)	37
@inp(n)	37
@fld(table.field)	37
@fmt(value,format)	37
@pbrk()	37
@plc(prompt,sessionvar,program)	38
@prg(ProgramName,InputArgument)	38
@row(TableName)	38
@rptsubtotal(TotalName,Format)	38
@time()	38
@tblsubtotal(Format)	39
@trksubtotal(TotalName,Value)	39
Macro Hierarchy	39
Using Embedded Reports	39
Integration with existing applications	39
Common Database Tables	40
Defaulting Inputs with Session Variables	40
Defaulting Inputs with 4GL Programs	42
Using the @prg() macro	42
Javascript based totaling and subtotaling	43
RptSubTotal(BreakBy, ColumnForSubTotal, NumberOfColumns, ID)	44
TrkSubTotal(BreakBy, ValueToAdd, ID)	45
SumSubTotal(ID)	45
CurSubTotal(ID)	45
Counter(CounterID)	45
RptCounter(CounterID, ID)	46
Sigma (SigmaID, ValueToSum)	46
RptSigma (SigmaID)	46

Min, Max, Count and other mathematics	46
<i>Planned Future Enhancements</i>	<i>47</i>

License

Written by Scott Auge scott_auge@yahoo.com sauge@amduus.com
Copyright (c) 2002 Amduus Information Works, Inc.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. All advertising materials mentioning features or use of this software must display the following acknowledgement:
This product includes software developed by Amduus Information Works Inc. and its contributors.
4. Neither the name of Amduus nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY AMDUUS AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AMDUUS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contact Scott Auge at sauge@amduus.com if you wish a different type of license.

What is it?

A report generator allows you to easily create reports with little to no programming required. One can quickly add a report for other users in the company to use. Since the report generator described herein is web based, you can make these reports part of your company's intranet portal.

Features available in the latest report generator:

- Addition of the @dur() macro which measures report generation time in seconds
- Ability to manage buffer/tables accessible to the report generator via a web interface

**The following are used to make Denkh useful as a content management system!
Now Denkh can drive your portal as well be a web based report generator!**

- One can use @emb(ReportName,) to run reports without inputs or queries.
- One can directly run a report without queries via HTMLRpt.html?....
- One can directly run a report without inputs via HTMLRpt.html?...
- Report no longer defaults <table> and </table> to contain record and summary sections.
- Place these in the report definitions for Page Header and Page Footer.

Previous features:

- @fmt() supports numeric, time, and string formats for the given parameter (ie Easy output formatting, regardless of data type)
- @tblsubtotal() can have private internal subtotal accumulators
- @drill() that allows the user to "drill" from one report into another via a hyperlink
- Column headings for each field can be specified for each report layout
- Side-labels for each field can be specified for each report layout
- Page totals, sub-totals & totals (without turning into a pretzel)
- Since it is on the web, Preview or print to screen with optional paper printing later on
- Open Source so you can modify look and feel, as well research "how it does it."
- Use HTML Editors to develop your report sections (then cut and paste the code in and out of the editor)
- Dynamic queries defined by the user
- Format reports using HTML/CSS2 for a more polished look
- Ability to include embedded reports within reports (a report repository)
- Ability to embed data fields from the query into the report
- Ability to embed page breaks into the report
- Ability to ask the report user questions for report parameters
- Ability to restrict which tables the report generator can reach

- Ability to look into Progress, Oracle, and ODBC based databases
- Web based so it can be reached easily by people (include in your portal!)
- Call out to Progress 4GL programs from within a report (good for integration)
- Use session variables within a report (good for integration)
- Include date and time within a report
- Include user's input in the output of the report
- Ability to access rowid of records in report (good for integration)
- PDF/MS Word documentation
- Report examples/Source Code examples
- Easy to use report listing system with search by name functionality

How does one start creating reports? Read on!

Basic Software Requirements

- Progress RDBMS or Progress DataServer to another database (Progress supported OS)
- Progress Webspeed or Amduus Blue Diamond
- Web Browser (Netscape, Internet Explorer, Opera, Konqueror, Mozilla 1.0+)
- Workshop/Webspeed Tools for your OS, or 4GL Development System if using Blue Diamond.

Amduus is a Progress reseller – we can sell you licenses or lease you Progress licenses if you need them.

Basic Report User Requirements

- Access to a JavaScript enabled web browser
- Ability to use a web application

Basic Report Designer Requirements

- Access to a JavaScript enabled browser
- Ability to code in HTML
- Ability to code with JavaScript (for more advanced reports)
- Ability to code Progress Query strings

Advanced Report Designer Requirements

- Basic Report Designer Requirements
- Knowledge of E4GL programming

Installing the software

1. Retrieve the latest build from Amduus Information Works, Inc. at www.amduus.com Files are packaged in a ZIP file. The zip file is named `Denkh.yyyy.ddd.hh.ss.zip` where `yyyy` is the year, `ddd` is the julian date (1-365), and `hh, ss` stand for hour minutes respectively.

2. Install the build in a directory of it's own

3. If your using Progress Webspeed, configure Progress Webspeed to use that directory with a broker. Here is a sample ubroker.properties definition:

```
[UBroker.WS.report]
  uuid=932.99.000.SGB:1ee77e:cf3bbe04fd:-8000
  srvrStartupParam=-p web/objects/web-disp.p -weblogerror -db sports2000 -H
localhost -S sports -db amduus -H localhost -S amduusdb
  srvrLogFile=/tmp/report.server.log
  controllingNameServer=NS1
  brokerLogFile=/tmp/report.broker.log
  environment=report
  userName=
  groupName=
  appserviceNameList=report
  description=Report Application
  portNumber=3070

PROPATH=/appl/Denkh/src:/appl/eqn/src:/appl/BlueDiamond/plussrc:/appl/rpg/src
  workDir=/appl/Denkh/src
```

You will need to compile it, so open up the broker to Develop mode.

4. If you are using Blue Diamond, then configure the application like any other application for Blue Diamond by including the install directory in the PROPATH of the Blue Diamond messenger. It also will need to be compiled.

5. Create a database for the Amduus.df file (and others if available) or load it into your current database. (Separate database is suggested as this schema changes!)

6. Set up your URL and access it. For example

`http://amduus2/online/portal/rptgen/DenkhReporter.html` is the home URL. Amduus2 is the name of the machine the software can be found at. `/online/` is the CGI directory used to find the Progress messenger/Blue Diamond portal. The PROPATH will require you to include `rptgen` to

find the Web Objects and finally DenkhReporter.html is the main administration screen available to generate reports.

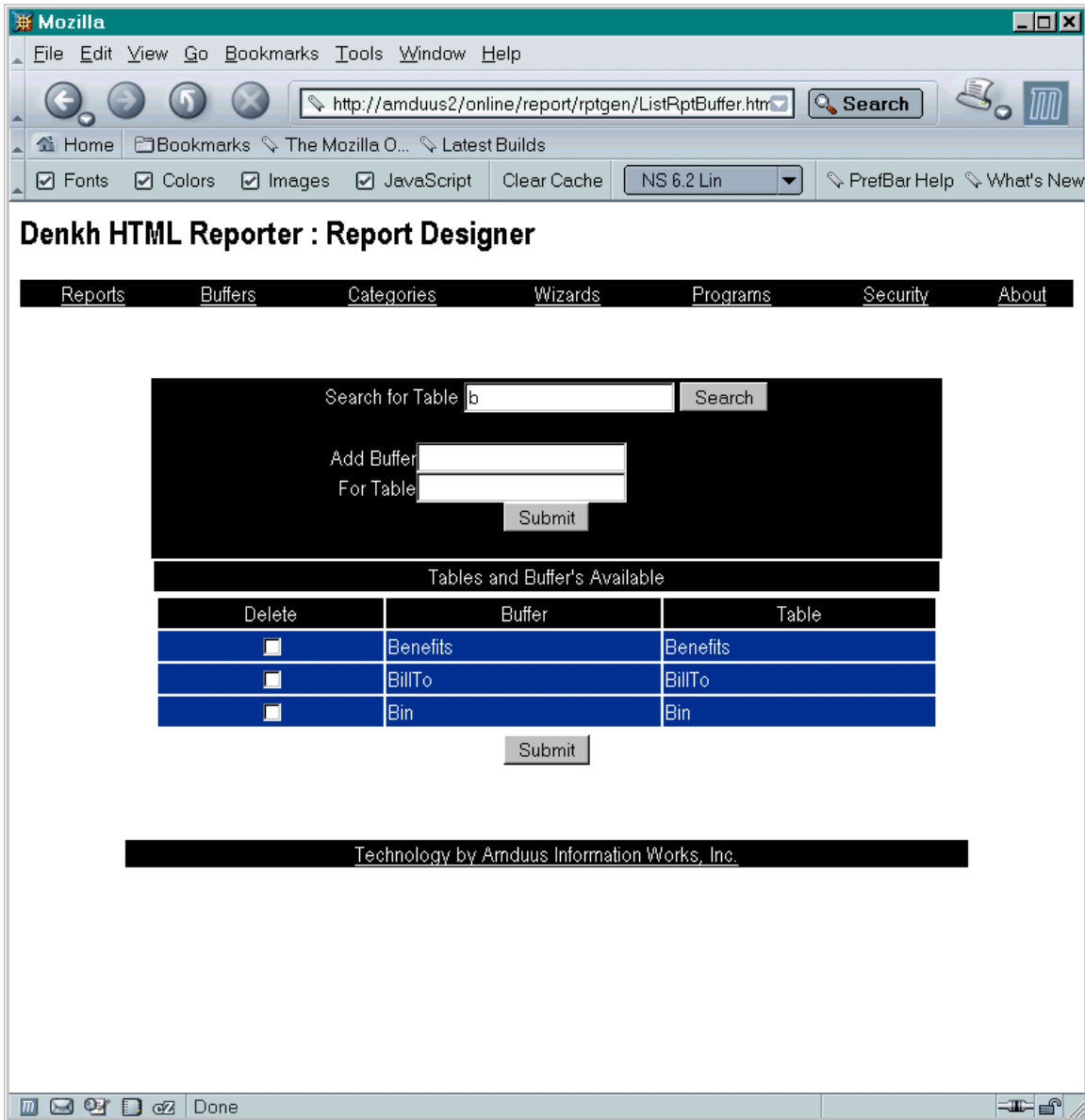
7. You will need to inform Denkh Reporter which tables/buffers it can use. For reports that may require different names for a given record (a buffer) of the same table, you can enter buffer names. Sometimes these names are clear than the table name.

For example, perhaps the table is named "ExtCust01". You could name a buffer "Customer" for table "ExtCust01" and reference the table as Customer in the report.

There are three ways to do this.

The first and easiest way is to make an data import file and use the data dictionary to load these values into the table RptBuffer.

The second is using the List Buffers page shown below:



You can reach this page from the AdminHTMLRpt.html page or by directly going to ListRptBuffer.html.

The page is simple to use. To add a table/buffer to the list of tables available to the reports, enter data into the Add Buffer [] For Table [] and press submit. You can name buffers the same as tables, but you cannot have two buffers of the same name.

To delete a table/buffer from being used in a report, click the check box beside the combination and press submit AT THE END of the listing.

The third way is programmatically:

You do this by running the PopRptBuffer.p program which will read in ALL your current working database's file names into the RptBuffer table.

If a table's name is not in RptBuffer, Denkh will not be able to recognize it to query on. You can also use the RptBuffer Editor in the Administration Screen to identify which tables to use for reporting.

You can do this by the following program in the script editor:

```
for each rptbuffer:
delete rptbuffer.
end.

create alias "DICTDB" for database sports2000.

run rptgen/PopRptBuffer.p.

for each rptbuffer:
disp rptbuffer.
end.
```

Before running this program, be sure the programs have been compiled against BOTH your database and the Amduus database (or if you combine them, the current database.)

8. Start setting up your reports with the administration screen.

Basic Components of Report

Reports are composed of five basic components. A page header, column headers (if applicable), a space for the records pulled up from the query, a summary area, a page footer, and of course the query it's self.

Under Denkh, the viewable components fall onto the web page in the following sequence.

Page Header
Column Headers
Record
Summary
Page Footer

All the displayable portions of the report can be formatted with HTML code mark-ups (including the use of stylesheets.)

Page Header

The page header is located at the top of the report, and follows an automatically placed <BODY> tag. You can include mark-ups such as TABLE, P, B, I, IMG, A, FONT, CSS. It is pretty flexible so you can create some beautifully formatted reports. This can be blank if you wish.

If you plan on advanced reports requiring the use of variables, you should define them in this section of the report.

Column Headers

If your report is going to be tabular, you can name the columns with comma delimited words.

For example: First Name,Last Name,E-Mail

When present, a row will be created on the table with columns named. This can be blank if you wish.

Record

For each row of the result set, this area of the report is rendered. If you have a simple tabular report, you may want to include <tr></tr> tags around your <td> tags.

If there is subtotalling, counting, grand-totalling, etc. going on, you will need to call out to those routines in this area. More about this is available later in the book.

If you have Column Labels, you should have that many TD tags here, otherwise you might get weird results on the screen.

Summary

This is the last <tr> of the default table composed of the column headers and record rows. This too should have as many <td> tags as columns to properly align up any summing, counts, or other mathematical operations performed. This can be blank if you wish.

Page Footer

Like the Page Header. This is merely between the <body> and </body> tags. This can be blank if you wish.

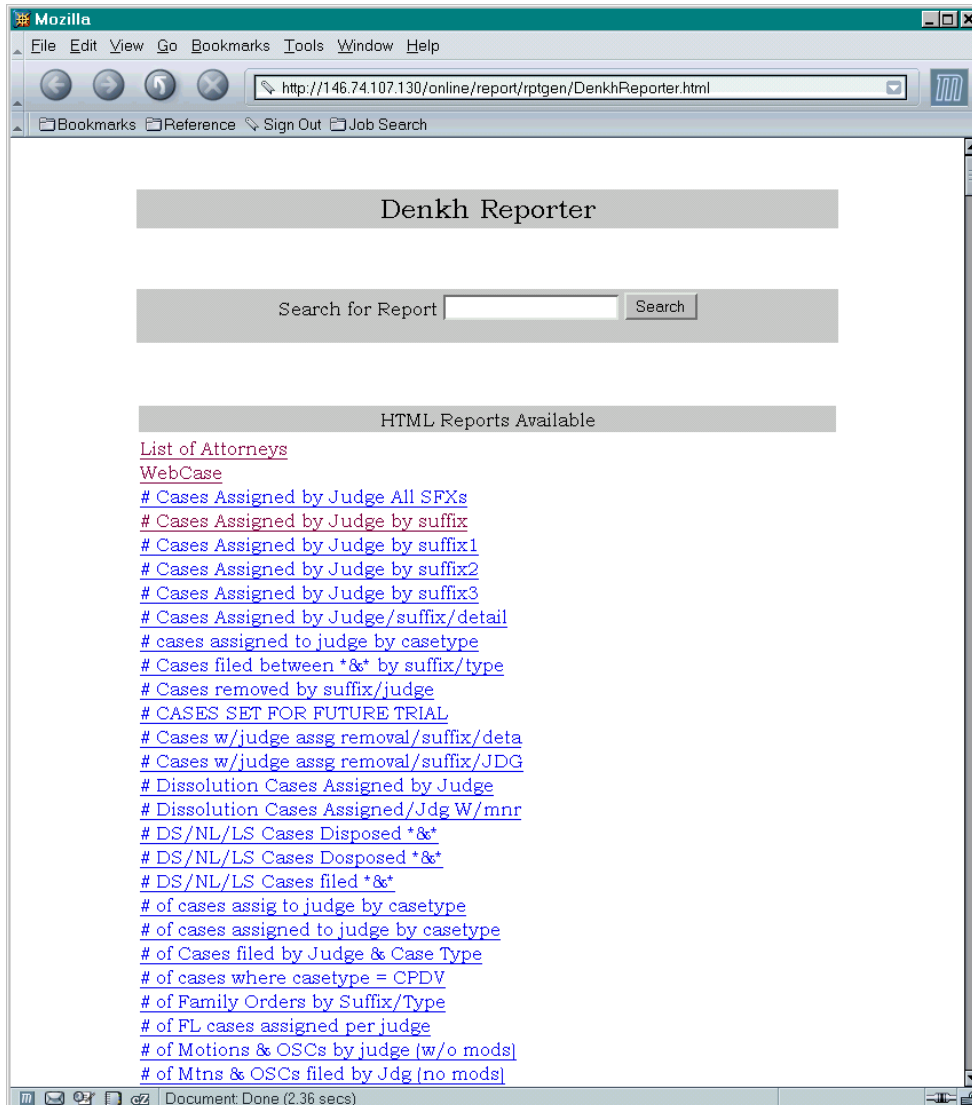
Query

This is a query that one would create for a dynamic query object in the Progress 4GL. One may want a programmer trained in the 4GL and the application's database tables available to craft the proper query to pull up the data.

NOTE: The Record and Summary areas do not include default <tr> tags. It is up to the report designer to include these in the report design.

Basic Running of a Report

The following are an example of the Denkh Reporter functioning in a major metropolitan court system within the United States.

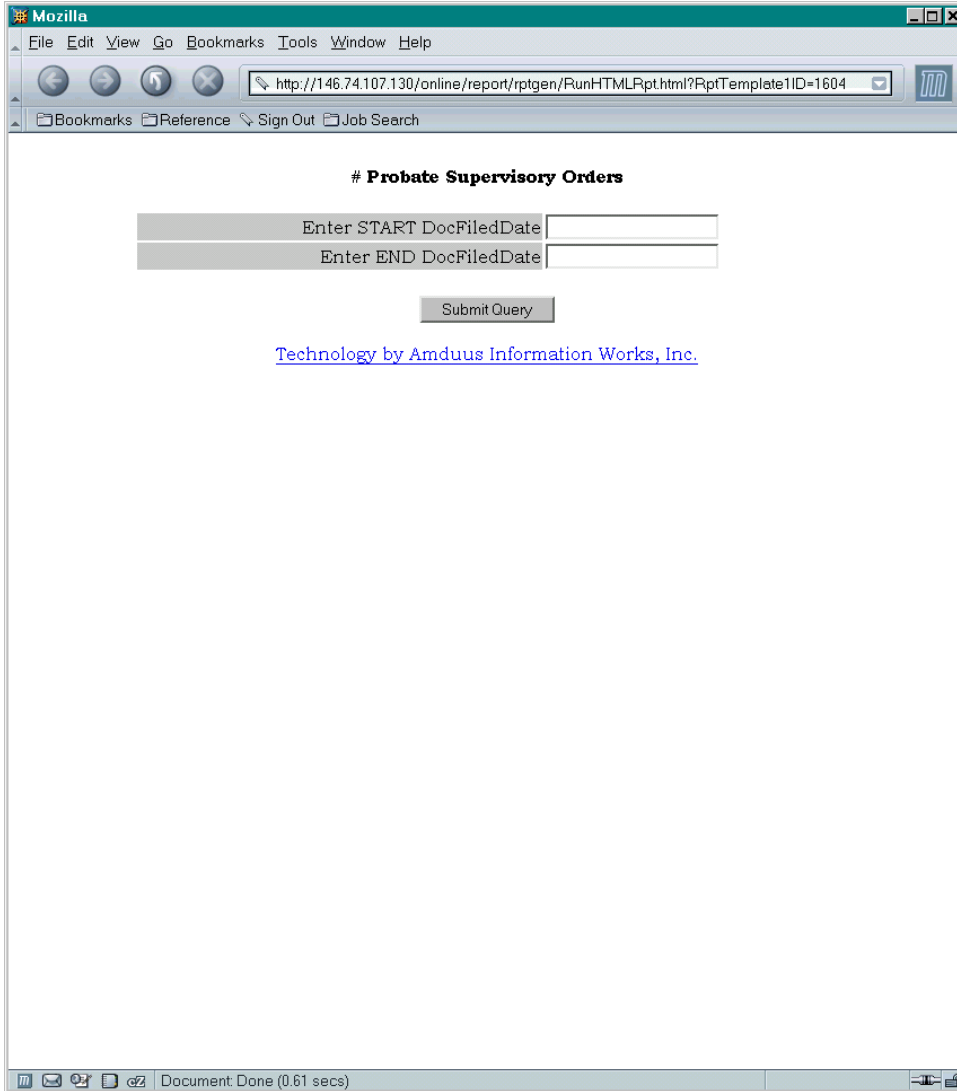


User Home page of the Denkh Reporter

One opens up the home page of the report generator using the URL to the page DenkhReporter.html. This page should be made available to the user community of the report generator. It will allow links to run reports, but no links to develop reports.

There is a search box to help bring up only those reports that contain that word or phrase. (It does not require a word index.) To bring up all the reports again, simply search on a blank box.

To use a report, one clicks on the title of the report, and a report parameter screen will appear:



The screenshot shows a Mozilla browser window with the address bar containing the URL: <http://146.74.107.130/online/report/rptgen/RunHTMLRpt.html?RptTemplate1ID=1604>. The browser's menu bar includes File, Edit, View, Go, Bookmarks, Tools, Window, and Help. The page content is titled "# Probate Supervisory Orders" and features a form with two input fields: "Enter START DocFiledDate" and "Enter END DocFiledDate". Below these fields is a "Submit Query" button. At the bottom of the page, there is a link: [Technology by Amduus Information Works, Inc.](#) The browser's status bar at the bottom indicates "Document Done (0.61 secs)".

These parameters are set up by the report designer. The report designer can set up as many parameters as they wish, as well make the prompts any set of words they wish (minus the use of parenthesis characters.)

The user would fill in the report parameters with their values and press submit.

The report would then be generated for them. Below is a tabular form of report. It has sub-totalling as well a summary of totals at the bottom (not shown.) The user would be free to import

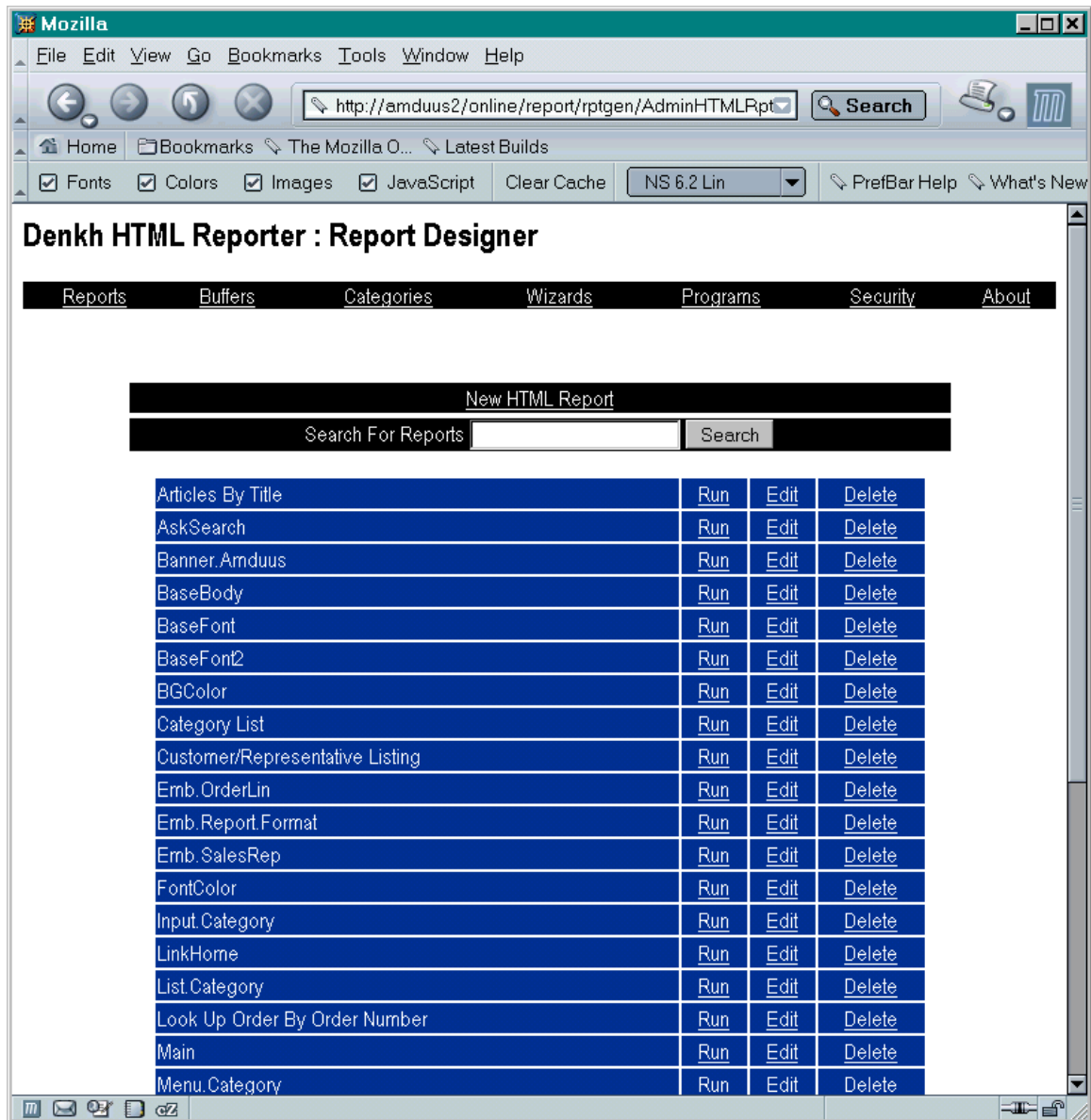
the data into an excel spreadsheet, or print it if they wish. Most newer browsers will allow the user to save the report and send it in an email message also.

All the column headers are definable by the report designer and the columns can be populated with data from the database. One can also see here the report includes sub-totaling information.

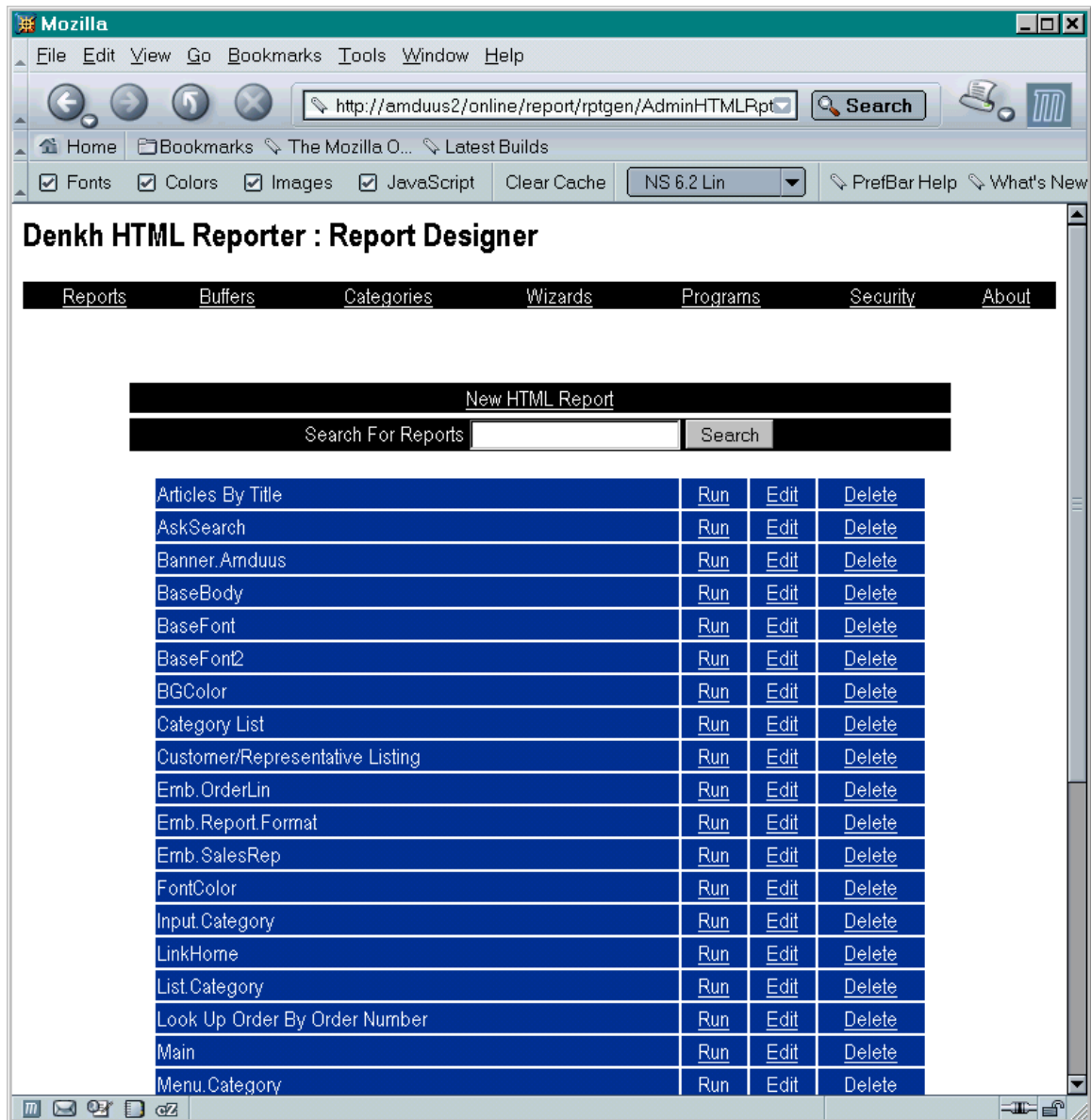
Case Number	Type	Case Filed	Filed	Code	Description	SubTotal
199PR144716	CN	03/03/99	01/05/00	D177	Order Changing Venue to:...	
180PR100859	CN	08/15/80	01/07/00	D180	Order [PR]	
182PR105497	CN	08/05/82	01/05/00	D180	Order [PR]	
186PR114690	CN	04/15/86	01/07/00	D180	Order [PR]	
193PR130864	CN	06/21/93	01/06/00	D194	Order Settling Current Acct/Report	
196PR139664	CN	10/11/96	01/07/00	D199	Order for Final Discharge	
						6
182PR104703	CNPR	04/13/82	01/07/00	D180	Order [PR]	
197PR142081	CNPR	12/29/97	01/07/00	D180	Order [PR]	
198PR142623	CNPR	03/12/98	01/06/00	D180	Order [PR]	
199PR144445	CNPR	01/22/99	01/03/00	D180	Order [PR]	
199PR145975	CNPR	09/30/99	01/11/00	D180	Order [PR]	
198PR144156	CNPR	12/15/98	01/05/00	D183	Order Appointing Probate Referee	
199PR143260	CNPR	05/28/99	01/07/00	D183	Order Appointing Probate Referee	
199PR143302	CNPR	05/28/99	01/07/00	D183	Order Appointing Probate Referee	
199PR146230	CNPR	11/23/99	01/04/00	D183	Order Appointing Probate Referee	
100PR146475	CNPR	01/04/00	01/04/00	D184	Order Appointing Temp Cons/Guardian	
100PR146479	CNPR	01/04/00	01/11/00	D184	Order Appointing Temp Cons/Guardian	
100PR146513	CNPR	01/05/00	01/05/00	D184	Order Appointing Temp Cons/Guardian	
100PR146553	CNPR	01/07/00	01/07/00	D184	Order Appointing Temp Cons/Guardian	
199PR146338	CNPR	11/24/99	01/11/00	D184	Order Appointing Temp Cons/Guardian	
198PR143154	CNPR	06/16/98	01/06/00	D194	Order Settling Current Acct/Report	
199PR144904	CNPR	05/05/99	01/10/00	D194	Order Settling Current Acct/Report	
193PR131625	CNPR	10/12/93	01/07/00	D199	Order for Final Discharge	
196PR137197	CNPR	04/05/96	01/07/00	D199	Order for Final Discharge	
198PR143394	CNPR	07/08/98	01/07/00	D199	Order for Final Discharge	
199PR145220	CNPR	06/21/99	01/07/00	D199	Order for Final Discharge	
						20
192PR128845	GR	08/07/92	01/06/00	D131	Order Terminating Guardianship	

Creating a report

To create a report, you will need to enter the AdminHTMLRpt.html screen. From this screen on can run reports, as well as administrate them.



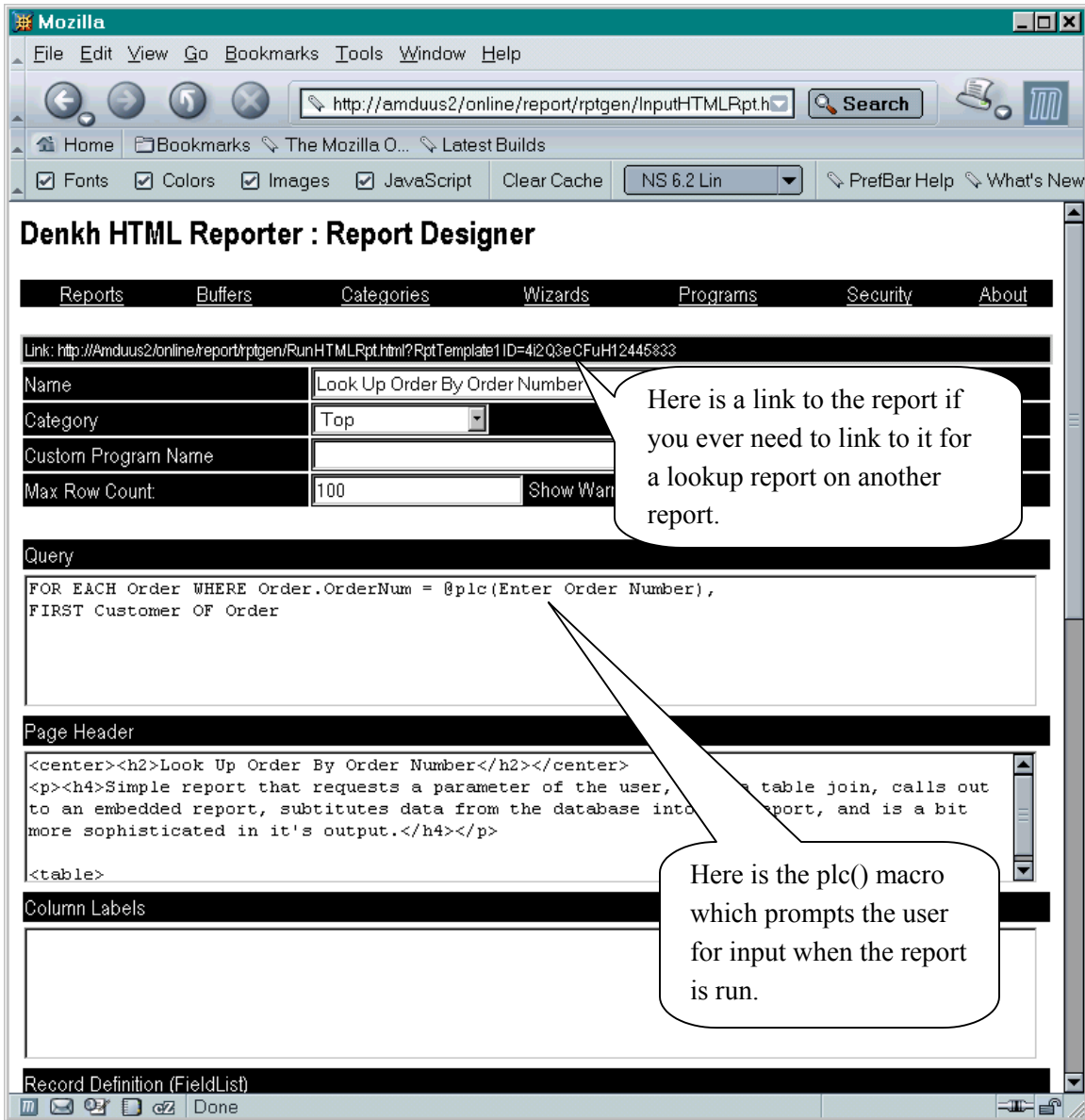
You will be presented with a screen to create new reports, or edit existing reports. Additional links are available to other features in the report designer. (More on that later in the book.)



Here we will click Edit on an existing report. All the components of the report are brought up in the design form of the report generator. This is a pretty advanced report, but will illustrate a lot – but not all - of the functionality the report generator can provide.

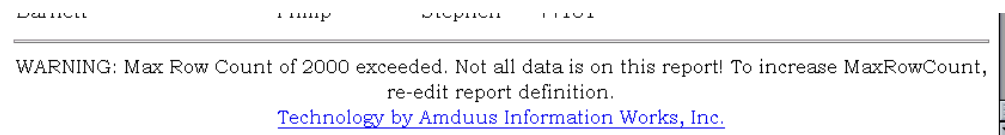
First one sees the title of the report as it will appear on the list of reports to the users. If you need to link to the report in another report (for a “lookup” to feed data into a input parameter), the link is available above the title.

As you can see, the queries can be quite complex. Within the queries are @plc() macros. These macros serve as place holders for what the user will enter in the form.



Report Design Form Part I

Since we could potentially send back too much information, we can limit the number of rows returned in the report. If the user needs more rows returned, the report should be updated here. A warning on the report will be presented to the user if not all the rows of their query are shown AND the Show Warning check box is checked:



Using the web for reporting has many advantages, but one is that a user requesting a very large report – one bigger than they could contend with - could “spin” a web agent (Progress talk, read the Webspeed or Blue Diamond manuals) so that it is not available to other users for a long, long time.

Following the Max Row Count is the Page Header definition. Here you can use the `@inp()`, `@time()`, and `@date()` macro's. See the reference in the back of the book for more information about these macro's.

In the Page Header section you should include the `<html><body>` tags.

Following the Page Header are the Column Headers. These will be turned into a set of `<tr><td></td></tr>` mark-ups by the report generator when available. If you have these, you should include a `<table>` tag as the last tag in your Page Header.

Inside the report template are areas the Record definition and Summary of the report. After the summary one should include a `</table>` marker in the Page Footer. The Summary area is “executed” after all the report sections have been exhausted from the record result set. This is so that any summary information will line up nicely in the report after the records have been listed.

Record Definition (FieldList)

```
<tr>
  <td>@fld(OrderLine.LineNum) </td>
  <td>@fld(Item.ItemName) </td>
  <td>@fld(OrderLine.Qty) </td>
  <td>@fmt(@fld(OrderLine.Price), $>>>9.99) </td>
  <td align="right">@calc(@fld(OrderLine.Qty) * @fld(OrderLine.Price), $>>>9.99)
    @trksubtotal(Grand, @calc(@fld(OrderLine.Qty) * @fld(OrderLine.Price), >>>9.99)) </td>
</tr>
```

Use the fld() macro here to match which database table.field should go here.

Summary Area

```
<td></td>
<td></td>
<td></td>
<td align="center" bgcolor="#CCCCCC">Grand:</td>
<td align="right">@rptstotal(Grand, $>>999.99) </td>
</tr>
```

Page Footer

```
</table>
```

These are special macros to aid in the totaling of your report. Each total can be named.

Report Design Form Part II

The record definition section of the report is the meat of the report to the user. It is exceptionally flexible to handle not only tabular reports, but letter reports also for easy merging of data with invoices, notices, marketing information, etc. Note there are many macro's you can call to manipulate information to place on the report. If there are no appropriate macro, you can create a 4GL program and call that via the @prg() macro to substitute in information.

Following is the summary area which is rendered after the final record has been found. Note with the Sub totaling tools, you will want the last sub-total to appear in the summary area.

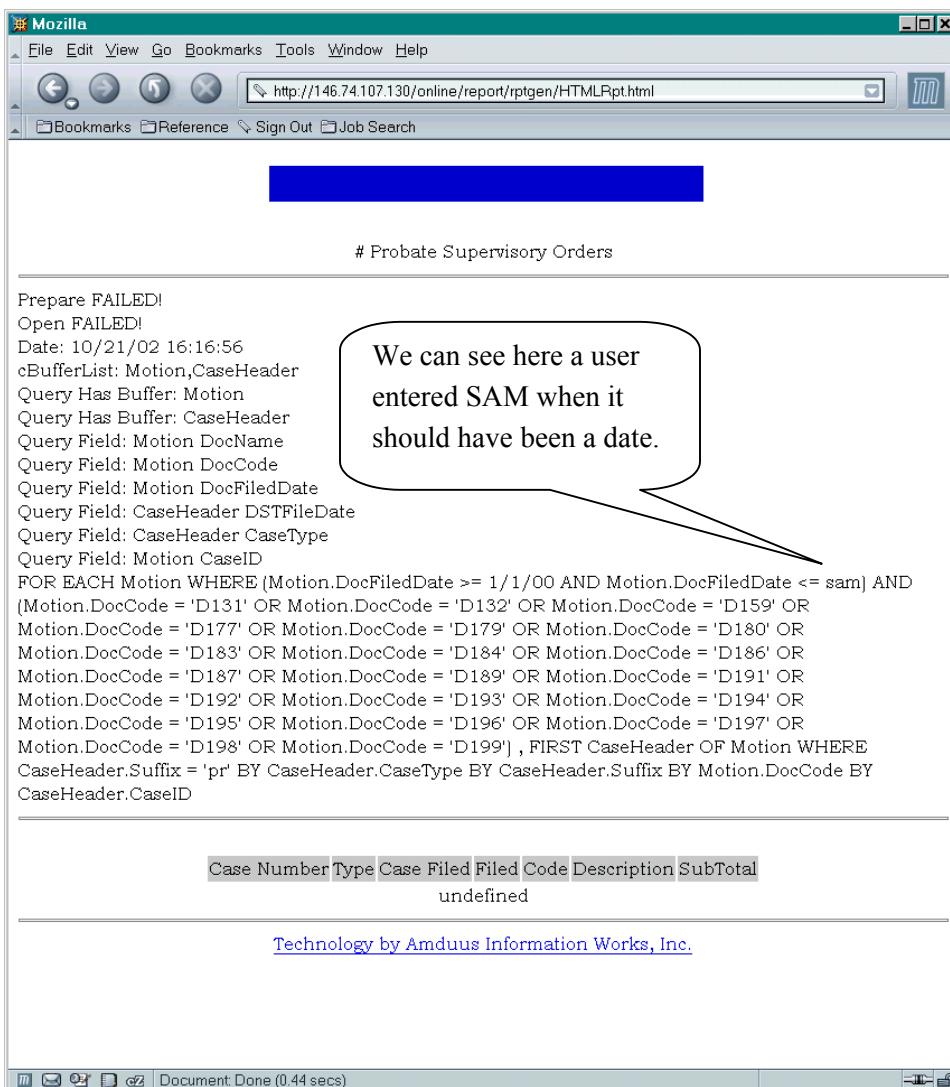
Finally is the Page Footer which is between the last `</table>` and the `</body>` tags should be placed report template.

Note: If you expect your report to be embedded – you should NOT include the `<html>`, `<body>`, `<header>`, etc. tags as it will simply be substituting in HTML code.

When things go wrong

When a report blows up, a screen of information becomes available to help determine what the problem is. It includes:

- The date and time of the blow up
- A listing of all the buffers the report thinks it needs
- A list of the buffers the report has allocated to the query
- A list of fields it thinks it needs from the database
- The query as sent to the database server



This information is usually sufficient to help explain what went wrong. By looking at the query, we can see the user entered an invalid piece of data, the word “sam” when it should have been a

date. Another place reports get burned is when a report designer requests a string, and forgets to put "" around the @plc() macro in the query.

Also available is the server log from the Webspeed agent. An example is given here:

```
S-0001>(Oct 21, 2002 16:16:56:425) [16013] sam must be a quoted constant or  
an unabbreviated, unambiguous buffer/field reference for buffers known to query  
. (7328)
```

```
S-0001>(Oct 21, 2002 16:16:56:426) [16013] QUERY-OPEN for query requires a  
previous QUERY-PREPARE. (7312)
```

Examples:

A more complex report

We want the report to display it's data in a more sophisticated manner – we want a line item listing for a given order. Then we want to display that inside another report that will include the header information for that order.

Once again, we go into the Editing screen for reports. What I did is design the table in an HTML editor for quick entry of the fields and table tags. I then cut and paste that into the input screen.

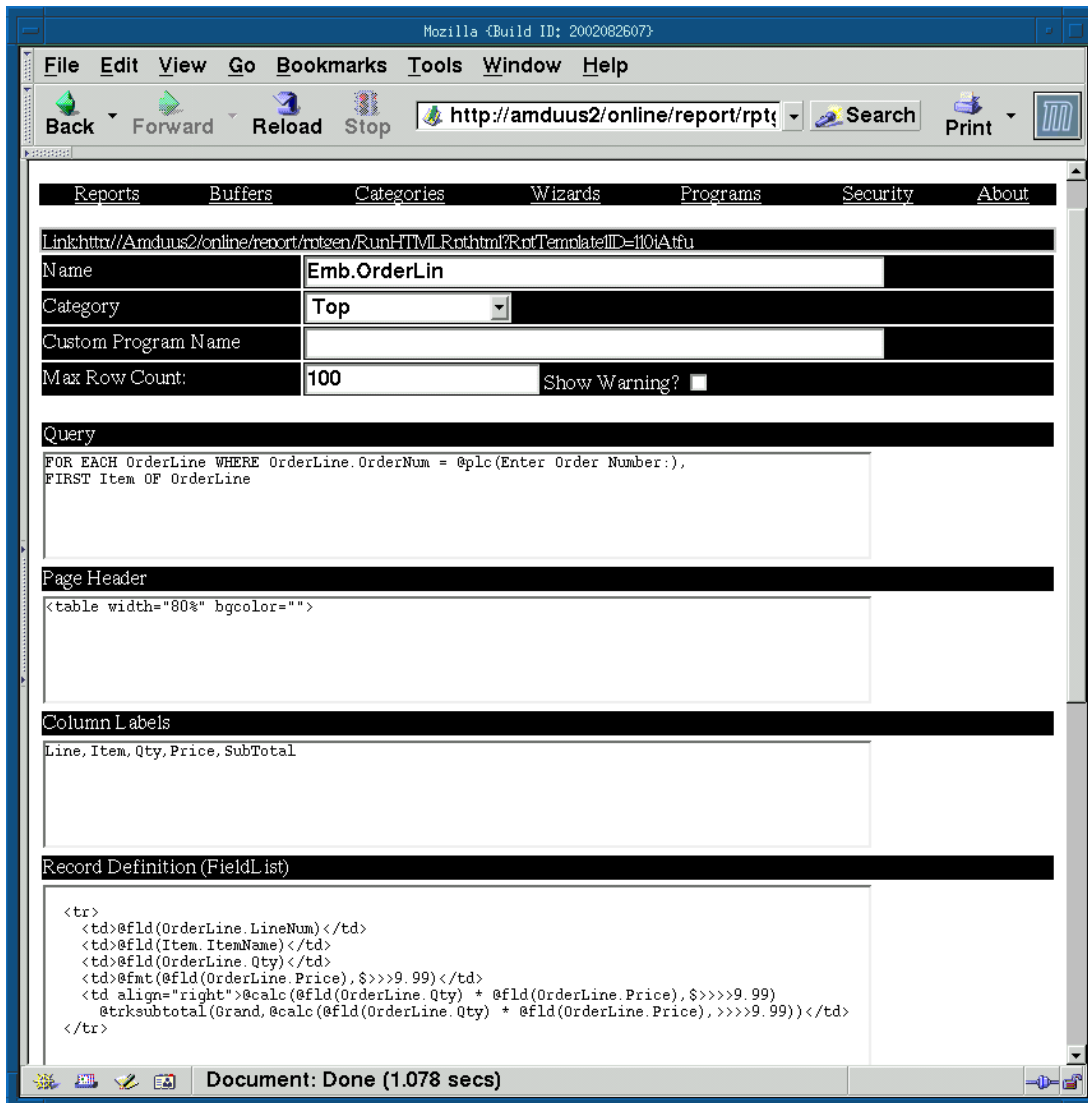


Figure 1 – Editing Screen for Report we are about to embed

Now lets try running the report. It should prompt for an order number and return a listing of the items on that order:

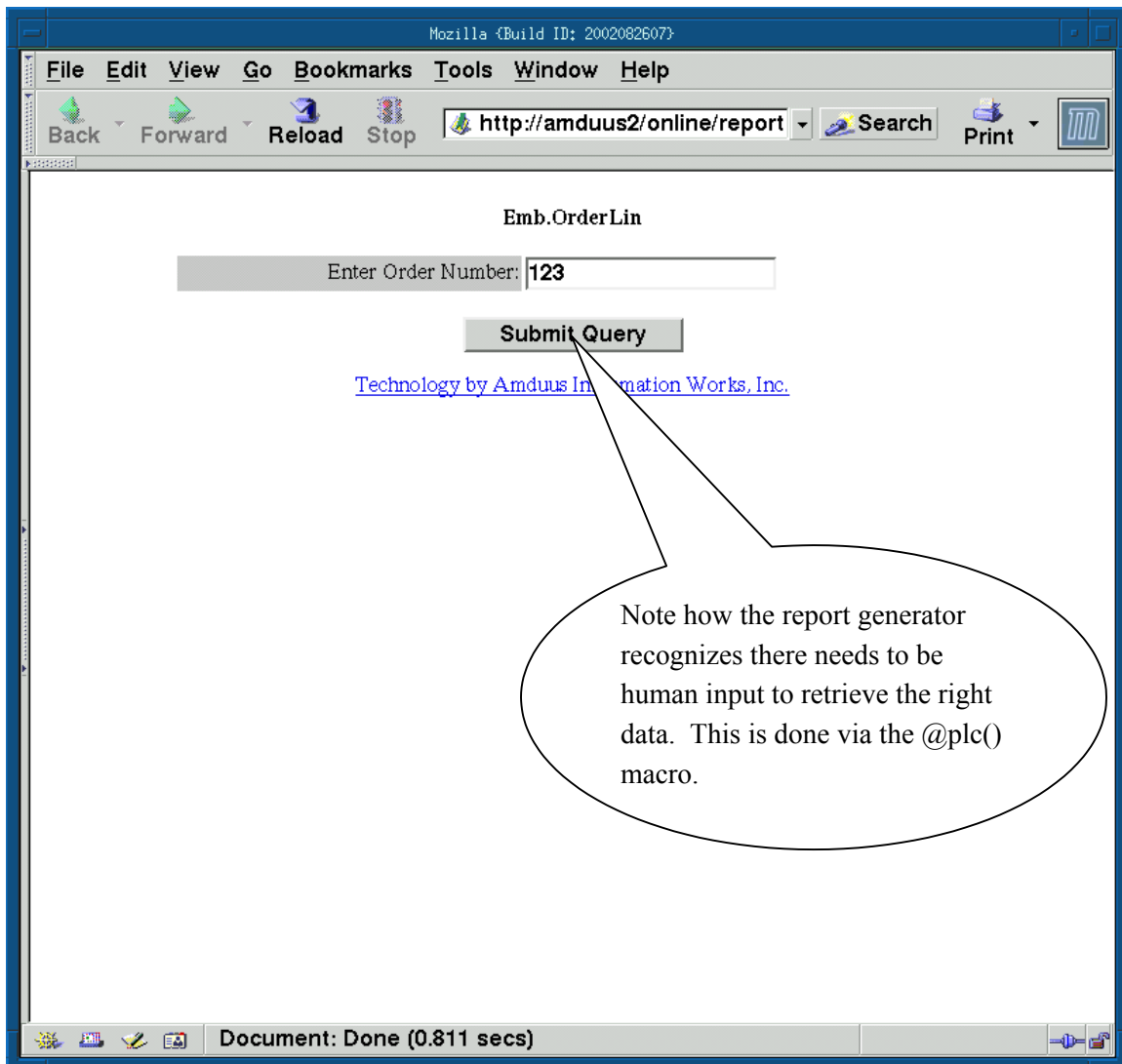


Figure 2 – Report Prompting User for Inputs

Line	Item	Qty	Price	SubTotal
1	Hockey Stick	20	\$37.00	\$740.00
2	Ski Wax - Red	64	\$2.75	\$176.00
Grand:				\$916.00

[Technology by Amduus Information Works, Inc.](#)

This is the data layout as defined in the Record section of the report definition. Note how the @fld() macro has been replaced with data from that table.field, as well how the @fmt(), @calc(), and @rpttotal() macros have been used.

Document: Done (0.521 secs)

Figure – Report Retrieving Data Based On User Input

Here is the report. The entire screen except for the Technology by Amduus... link is designed by the report designer.

Now we want to embed this report into another.

Note: If the report is already done – it is an abstraction away from the database layer for report designers. Embedded reports can act as building blocks for designers to put together – all they need to know are the report's input parameters – not the table names or how to join them!

The screenshot shows a Mozilla browser window displaying an HTML report. The report is structured as follows:

- Page Header:**

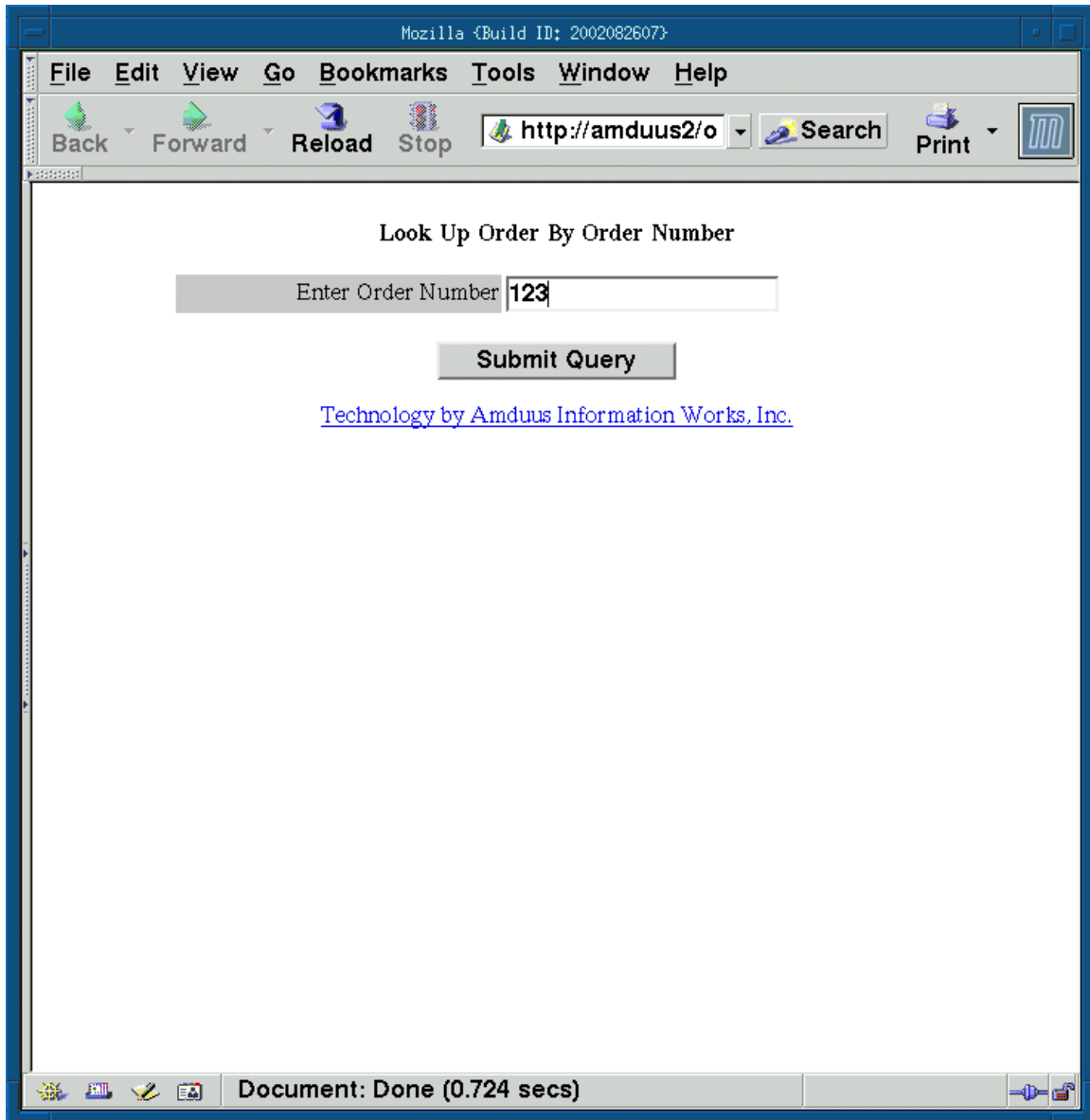
```
<center><h2>Look Up Order By Order Number</h2></center>
<p><h4>Simple report that requests a parameter of the user, does a table join, calls out to an embedded report, substitutes data from the database into the report, and is a bit more sophisticated in it's output.</h4></p>
<table>
```
- Column Labels:** (Empty table)
- Record Definition (FieldList):**

```
Number:</font></div></td>
<td>@fld(Order.OrderNum)</td>
<td bgcolor="#CCCCCC"><div align="right">
color="#FFFF00"></font></div></td>
<td>@fld(Order.PO)</td>
<td bgcolor="#CCCCCC"><div align="right">
Date:</font></div></td>
<td>@fld(Order.PromiseDate)</td>
</tr>
<tr>
<td bgcolor="#CCCCCC"><div align="right">
Name:</font></div></td>
<td colspan="3">@fld(Customer.Name)</td>
<td bgcolor="#CCCCCC"><div align="right">
Status:</font></div></td>
<td>@fld(Order.OrderStatus)</td>
</tr>
</table>
<br>
<center>@emb(Emb.OrderLin,@fld(Order.OrderNum))</center>
```
- Summary Area:** (Empty table)
- Page Footer:** Document: Done (0.893 secs)

A speech bubble points to the `@emb()` macro call in the Record Definition section, with the text: "Here you can see the @emb() macro call out to the report designed above to include it's output as part of this report."

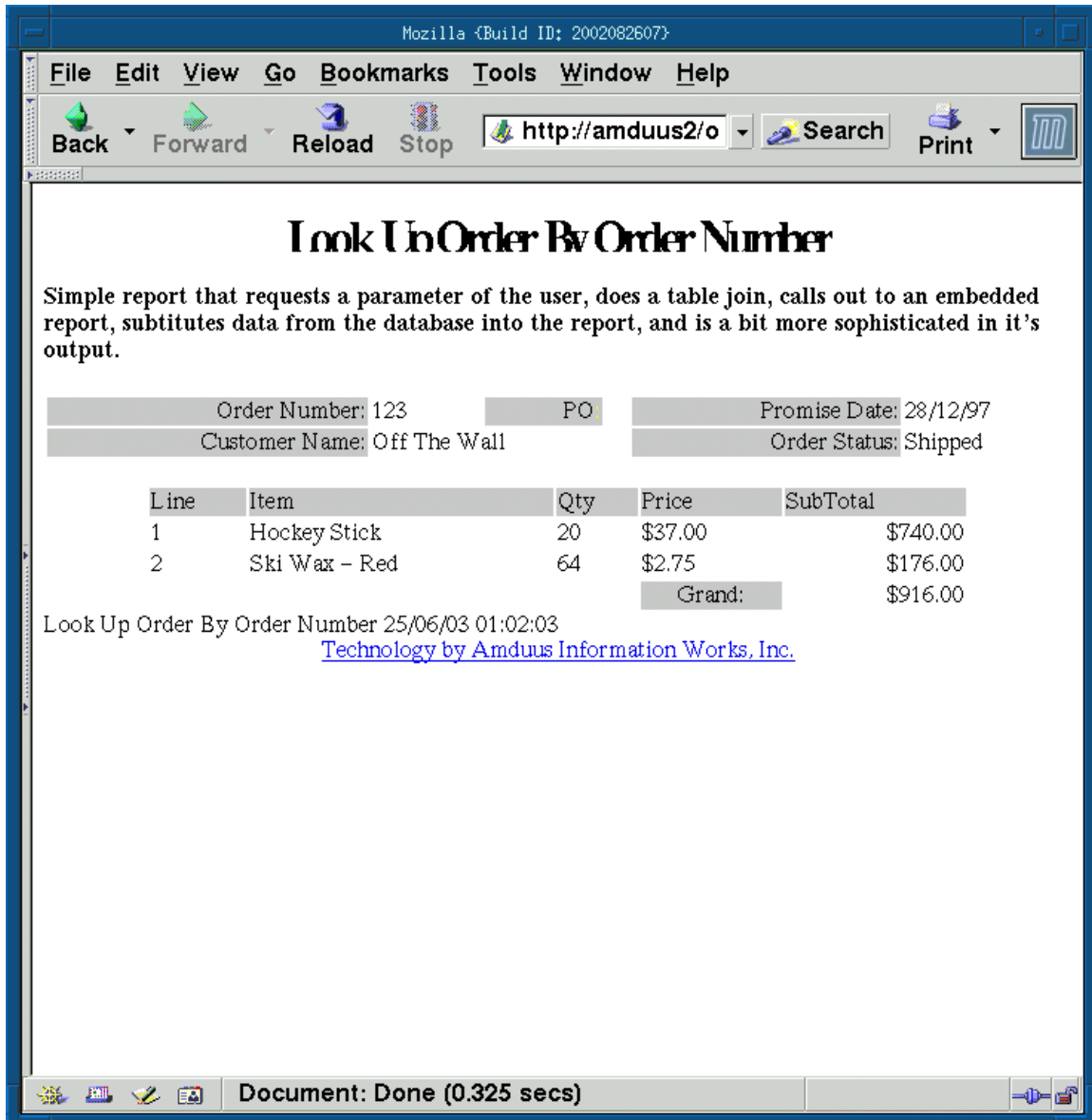
A report using the order lines report as part of it's output

Here we discover the power of embedded reports. All the report designer needs to do is call out to the above designed report with the `@emb()` macro. Denkh HTML Reporter will substitute the macro with the resulting report.



Running the report with an embedded report

Here we are running the report using the embedded report. It will prompt for an order number, create a header for that order, then call out to the embedded report that states the lines of that order and place that in the output.



Mozilla (Build ID: 2002082607)

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop http://amduus2/o Search Print

Look Up Order By Order Number

Simple report that requests a parameter of the user, does a table join, calls out to an embedded report, substitutes data from the database into the report, and is a bit more sophisticated in it's output.

Order Number: 123	PO	Promise Date: 28/12/97
Customer Name: Off The Wall		Order Status: Shipped

Line	Item	Qty	Price	SubTotal
1	Hockey Stick	20	\$37.00	\$740.00
2	Ski Wax - Red	64	\$2.75	\$176.00
	Grand:			\$916.00

Look Up Order By Order Number 25/06/03 01:02:03
[Technology by Amduus Information Works, Inc.](#)

Document: Done (0.325 secs)

A report using the results of an embedded report!

Wha-la! The end report. See how using embedded reports takes the report designer away from the database tables needed. One could also create an embedded report for the given order's header information.

Embedded reports can contain embedded reports. (Be careful not to get into an embedded report infinite loop.)

A letter report

Below is the record definition for a letter report. As a groomed HTML coder will realize, it is a bit of an awful mess (imported from another reporting program that was line printer based.) But the report generator can work with it.

As you can see from the highlighting, there are images for letter head, and more advanced formatting. In addition, forced page breaks can be made when printing from the browser. This is a two page letter that is sent out to n people depending on the input parameters.

```

<tr><center></center><br><br>
<b>      @fld(ThisSite.SiteName), @fld(ThisSite.County)<br>
          @fld(FacilitySpec.Add2)<br>
          @fld(FacilitySpec.City), @fld(FacilitySpec.State)
@fld(FacilitySpec.Zip)</b><br>
<br>
<br>
<br>
<br>
<br>
<br>
To: @fld(Recipient.ToName)<br>
    @fld(Recipient.ToAddr1) @fld(Recipient.ToAddr2)<br>
    @fld(Recipient.ToCity), @fld(Recipient.ToState) @fld(Recipient.ToZip)<br>
<br>
<br>
<br>
<br>
RE: @fld(CaseHeader.CaseTitle)          Case Nbr: @fld(CaseHeader.CaseID)
<br>
<br>
          NOTICE OF CASE MANAGEMENT CONFERENCE <br>
          <br>
          THE ABOVE ENTITLED CASE HAS BEEN SCHEDULED FOR A CASE MANAGEMENT CONFERENCE
<br>
          AS FOLLOWS: <br>
          <br>
          DATE: @INP(2)   TIME: @INP(3)   IN: @fld(FacilitySpec.FacilityDesc)<br>
          LOCATION: @fld(FacilitySpec.Add1), @fld(FacilitySpec.Add2)          <br>
                   @fld(FacilitySpec.City), @fld(FacilitySpec.State)
@fld(FacilitySpec.Zip)<br>
<br>
          IF THE ABOVE DATE PRESENTS A SCHEDULING CONFLICT FOR COUNSEL, OR IF<br>
          YOU WISH FURTHER INFORMATION, CONTACT THE DOMESTIC CALENDAR OFFICE<br>
          AT CENSORED OR CENSORED.<br>
          <br>
          SETTLEMENT CONFERENCE AND TRIAL DATES WILL BE SET AT THE TIME OF<br>
          THE CASE MANGEMENT CONFERENCE.<br>
<br>
          YOU ARE EXPECTED TO BRING YOUR CALENDAR TO COURT.<br>

```


<h5>If you, a party represented by you, or a witness to be called on behalf of that party need an accommodation under
the American with Disabilities Act, please contact the Court Administrator's Office at CENSORED, or use the Court's
TDD line, CENSORED.</h5>

Date Issued: @inp(1)

<p style="page-break-before: always">

CASE MANAGMENT CONFERENCE

INSTRUCTION SHEET

PLEASE TAKE NOTICE: You are required to appear at a Case Management Conference
(CMC) on the court date stated on the attached Notice of CMC.

INITIAL CASE MANAGEMENT CONFERENCE

If this is the Initial Case Management Conference in this matter, you
must complete Part 1 of the attached CMC Questionnaire, mail completed copies
to all attorneys or self represented parties in the case, and file the original
with the Court at least 15 calendar days before the scheduled court date.

TRIAL/REVIEW CASE MANAGEMENT CONFERENCE

If this is a Trial/Review Case Management Conference, you must complete
Part 2 of the attached CMC Questionnaire, mail copies to all attorneys or self
represented parties in the case, and file the original with the Court at least
15 calendar days before the scheduled court date.

TELEPHONIC APPEARANCE

If you want to appear at a Case Management Conference by telephone,
complete the attached Request for Telephone Appearance Form, file or fax it
to the Court between 6 and 10 court days prior to the CMS. You will be
contacted about the scheduling of the conference call. The Court may require
a personal appearance of all counsel and parties at any CMC. This telephonic
appearance is not available to parties eligible for the early neutral
evaluation program (see enclosed notice).

ANY FAILURE TO COMPLY WITH ANY OF THESE PROCEDURES OR A FAILURE TO APPEAR AT A


```

CASE MANAGEMENT CONFERENCE MAY RESULT IN SANCTIONS AGAINST YOU.</b><br>
<br>
Estas instrucciones son obtenibles en espanol en las oficina del actuuario<br>
(clerk's office), CENSORED.<br>
<br>
<br>
-----<br>
<br>
Parties and/or Attorneys of Record:<br>
<br>
@fld(Recipient.CCName)<br>
<br>
<h5>DECLARATION OF SERVICE BY MAIL: I declare under penalty of perjury that I
served this notice by enclosing a true copy in a sealed<br>
envelope, addressed to each person whose name is shown above, and by depositing
the envelope with postage fully prepaid, in the<br>
U.S. Mail at CENSORED on @INP(4). CENSORED, Chief Executive Officer/Clerk by
CENSORED, Deputy.</h5><br>
<br>
<p style="page-break-before: always"></tr>

```

More advanced subjects

Calculations

Simple Calculations can be performed on data in the report via the @calc() macro. The first argument to the macro is the formula to be calculated, the next is the format the results should be returned in.

The operations +, -, *, and / can be performed on integers and decimals in an algebraic expression. For example:

```
@calc(2*(3+4),99)
```

The calc macro takes place after the fld and prg macros have been executed, so it can use these as a source of values. Example:

```
@calc(@fld(OrderLine.Qty) * @fld(OrderLine.Price),>99.99)
```

Presently the @calc() macro cannot be nested.

The @calc() macro is performed before the @trksubtotal() macro, so something like:

```
@trksubtotal(GrandTotal,@calc(@fld(OrderLine.Qty) * @fld(OrderLine.Price),>9))
```

is possible.

Other JavaScript available mathematical operations and functions are available.

Totaling and sub-totaling

This is achieved by using the macro's `@trksubtotal()`, `@rptsubtotal()`, and `@tblsubtotal()`.

The `@trksubtotal` macro will update the internal accumulator for the named subtotal. It should be called before the `@rptsubtotal` or `@tblsubtotal` macros if the current query row returned should be included.

The `@rptsubtotal()` macro simply returns the subtotal in the requested format.

The `@tblsubtotal()` returns a table of subtotals in the requested format.

See also the section "Javascript for totaling and subtotaling."

Page Breaking

One can use the CSS2 styles if the browser supports it. This is available in Internet Explorer 4.x as well Mozilla 1.0+/Netscape 7.0+.

At the page break, place the line:

```
<p style="page-break-before: always">
```

Before the content for the next page. This is useful for running multiple pages of letters. A good place for this is in the Page Footer section.

Using Hyperlinks in the @plc() macro's for "Lookups."

One can use hyperlinks in the `@plc()` macro's to make "lookup screens" for inputs. In the report definition screen there will be links to the given report. Merely make the HREF this value. *Just be careful not to use ANY (or) in the hyperlink. The parser does not handle parenthesis between plc parenthesis very well.*

Example:

```
@plc(<a href="RunHTMLRpt.html?RptTemplateID=fghdkk337wihjd"
target="_blank">Enter Responder Code:</a>)
```

Drilling into other reports

It is possible to drill from the results of one report into another report with the @drill() macro. The drill macro provides a hyperlink into another report by it's name. The report developer has the ability to determine what the hyperlink says. The macro provides inputs to the report just as if the user had entered them, but the inputs come from constants or other macros. The macro provides the ability of the drilled down report to open a new window, act in the parent window, or in the current window.

Macros

<i>@calc(Formula,Format)</i>
Used in: Record
Perform a calculation. Handles Algebraic Formulas. Returns the value in Format format. See the FORMAT Phrase in the 4GL reference for appropriate descriptions.
Example: $\text{@calc}(\text{@fld}(\text{OrderLine.Qty}) * \text{@fld}(\text{OrderLine.Price}), \$>>>99.9)$ is $\text{OrderLine.Qty} * \text{OrderLine.Price}$ Example: $\text{calc}(\text{@fld}(\text{OrderLine.Qty}) * (\text{@fld}(\text{OrderLine.Price}) - (\text{@fld}(\text{OrderLine.Discount}) / 100 * \text{@fld}(\text{Orderline.Price}))), \$>>999.99)$ is $\text{OrderLine.Qty} * (\text{OrderLine.Price} - (\text{Orderline.Discount} / 100 * \text{OrderLine.Price}))$ And of course one can use decimal/integers such as @calc(2*(3+4),999).

<i>@date()</i>
Used in: Page Header, Page Footer, Record
Places date of report run on the page. No arguments to the macro. Example: @date() Format is Progress parameter definition.

<i>@drill(NameOfReport,Arg1 Arg2,Target,HyperLinkText)</i>
Used in: Page Header, Page Footer, Summary, Record
Creates a hyperlink with text from the HyperLink Text argument to NameOfReport report with arguments Argn in window Target (_blank, _parent, ...) Arg should be in order of @plc() in the query of the target report. For Example:

```
@drill(Look Up Order By Order Number,@fld(Order.OrderNum),_blank,@fld(Order.OrderNum))
```

@dur()

Used in: Page Footer

Reports the length of time it took to generate the report/subreport. For Example:

Duration of report in seconds: @dur()

@emb(ReportName,ReportArg1|ReportArg2|...)

Used in: Page Header, Page Footer, Record

Allows embedding of a report into another report. There can be multiple reports embedded, and embedded reports can refer to other embedded reports. Embedded reports should not exceed 32KB of output. This restriction does not reflect on the primary report however. Example:

@emb(InvoiceLines,@fld(InvoiceHeader.InvoiceNumber)|12/10/99) calls the report "InvoiceLines" with two arguments, the field value of "InvoiceHeader.InvoiceNumber" and "12/10/99."

@inp(n)

Used in: Header, Footer, Record

Single Argument of the position number of the input the user gave to a plc(). Example: @inp(1) will return the input to the first @plc() in the query.

@fld(table.field)

Used in: Record

Single Argument of table and field to appear in the location of the macro. Example:

@fld(WebState.Name)

@fmt(value,format)

Used in: Record

Format a value according to the format phrase "format."

Example: @fmt(fld(OrderLine.Price),\$>>>9.99) or @fmt(12345,HH:MM:SS AM)

@pbrk()

Used in: Record,Page Header, Page Footer

Inserts a page break for CSS2 compliant browsers (Mozilla 1.0+, Internet Explorer 5.0+ Netscape 7.0+, others...)

@plc(prompt,sessionvar,program)

Used in: Query

The first argument is the text to appear on the user input page. For example: @plc(First Name:) will present `First Name:` on the screen asking for parameters for the report. The second and third inputs refer to defaulting values for the input. The second argument refers to a Session variable held in the Web State table. (See the section Session Variables.) The third argument refers to a program to run receive the default input with. (See the section 4GL Programs.)

Just be careful not to use ANY (or) in the prompt. The parser does not handle parenthesis between plc parenthesis very well.

@prg(ProgramName,InputArgument)

Used in: Record, Page Header, Page Footer

Allows the report to call out to a specialized progress routine. The progress routine must have one input of type character and one output of type character. If your program needs more than one input, you should combine them into a string and slice and dice in your program. Macro is replaced by the string returned by the program.

@row(TableName)

Used in: Record

Returns the rowid for the given record in the query. This is a way to send the record to a @prg() macro to aid in record navigation, as well a @emb() macro.

@rptsubtotal(TotalName,Format)

Used in: Record, Summary

Returns the subtotal named TotalName in the format Format. See the FORMAT Phrase in the 4GL reference for appropriate descriptions.

Example: @rptsubtotal(@fld(Order.OrderNumber),>>>9.99)

@time()

Used in: Header, Footer, Record

Returns the time the report was run in HH:MM:SS

<i>@tblsubtotal(Format)</i>
Used in: Summary
Returns a table of sub-totals in format Format. See the FORMAT Phrase in the 4GL reference for appropriate descriptions. Example: @tblsubtotal(>>>>9.99)

<i>@trksubtotal(TotalName,Value)</i>
Used in: Record, Summary
Adds Value to the running total named TotalName. Value can be integer or decimal. Example: @trksubtotal(@fld(Order.OrderNumber),@fld(OrderLine.LineTotal))

Macro Hierarchy

Macro's used in report output are evaluated in the following order:

1. @fld()
2. @row()
3. @date(), @time()
4. @inp()
5. @prg()
6. @calc()
7. @trksubtotal()
8. @rptsubtotal(), @tblsubtotal()
9. @emb()

Using Embedded Reports

Often one can make reports in "pieces." Reports may have common information displayed in a common fashion. Embedded reports makes it possible to have a repository of "sub-reports" one can create other reports out of.

Embedded reports can call into other embedded reports.

Be sure an embedded report does not return more than 32K of output.

Integration with existing applications

One uses a report program to further enhance an existing program with output not currently available from the existing program. The following section discusses how to integrate the report program with existing applications from such vendors as QAD, Astea, and home brewed code.

Common Database Tables

The most common and easiest integration point is to simply link the report generator to the same data source as the application it should provide reports for. One then creates reports based on the data dictionary and data found in the database.

Defaulting Inputs with Session Variables

There will be times when the programmer or installer will need to integrate the report writer directly into the application so it can work with the same state data the application is. A means to achieve this is by using session variables.

The basic steps are as follows:

- Create the hook calling into the report program with NVP containing the variable information you wish to pass into the report program.
- Edit the DenkhReporterOuputHeaders.i program to include code to store away these variables into session saving infrastructure for the report writer.
- Use these session vars to default inputs the user may need to enter with the plc() macro, or programmatically using a program specified in the plc() macro.

Lets look at this in more detail.

Here is a GUI based application that includes a menu item that calls out to the report program. When the user is working the GUI, they have a certain set of data set up as the “current working data.” When they click over into the report program, they want that information to be passed along too.

```
ON CHOOSE OF MENU-ITEM mWebReports
DO:
    DEF VAR cURL AS CHARACTER NO-UNDO.
    DEF VAR cURLCaseID AS CHARACTER NO-UNDO.
    ASSIGN cURLCaseID = cCaseID.
    ASSIGN cURL="~"C:\Program Files\Internet Explorer\IEXPLORE.EXE~"
    http://146.74.107.130/online/report/rptgen/DenkhReporter.html?CaseID=" + cURLCaseID.

    OS-COMMAND NO-WAIT VALUE (cURL) .
END.
```

Example code from an integration point in an application.

As you can read, you see that some NVP (Name/Value Pairs) are passed along into the report program. The DenkhReporter.html program includes this in the base release:

```

/* This output headers is made available to store cookies or do processing */
/* in a more centralized place when integrating the Denkh reporter to a    */
/* different application.  By calling the URL with your own NVPs, you can  */
/* set session information in Denkh for these values by parsing them in    */
/* DenkhReporterOutputHeaders.i.  Be sure your version of this file is in */
/* a propath before this directory.  That is, we recommend having:        */
/* ../Denkh/custom/  <-- Your custom code                                */
/* .. Denkh/src/     <-- Base code install                                */

PROCEDURE OUTPUT-HEADERS:
  {DenkhReporterOutputHeaders.i}
END.

```

Code from base Denkh Reporter used as an integration point

You should use the DenkhReporterOutputHeaders.i file to customize your NVPs into session vars. **Do not change the DenkhReporter.html program to do this – it may change in future versions and you will need to repeat your changes. Use the DenkhReporterOutputHeaders.i file in a different directory noted in the PROPATH before the base release propath for customized programming.**

Below is example code of DenkhReporterOutputHeaders.i used to pull the current working case id when the report software is called from the GUI app.

```

/* DenkhReporterOutputHeaders.i */
/* This file should be included in DenkhReporter.html.  */
/* Assume it is in the OUTPUT-HEADERS procedure.        */

DEF VAR RCSVersion1 AS CHARACTER INIT "$Header$" NO-UNDO.

DEF VAR cSessionID AS CHARACTER NO-UNDO.
DEF VAR cCaseID    AS CHARACTER NO-UNDO.
DEF VAR cError     AS CHARACTER NO-UNDO.

/* Create a quick session identifier and place into a cookie */

ASSIGN cSessionID = MakeID3(10).
SET-COOKIE("DenkhSession", cSessionID, ?, ?, ?, ?, ?).

/* When the reporter is called with a case, we want this case information to make */
/* it into this session of use so we can pre-populate stuff.                      */

ASSIGN cCaseID = GET-VALUE("CaseID").

RUN WriteState.p
(INPUT cSessionID,
 INPUT "Var",
 INPUT "CaseID",
 INPUT cCaseID,

```

```
OUTPUT cError).
```

Example Custom DenkhReporterOutputHeaders.i

You can use the plc macro to default this value into questions prompted of the user: As you can see, in the WriteState.p program, the data is named “CaseID.” (Note the category name of the data is called “Var”. You must always use “Var” as the category name for Denkh macros using this data to work.)

When running a report with the following macro:

```
@plc(Enter Case Number:,CaseID)
```

the user’s input screen will have a default value of the NVP stored in CaseID.

One can pull up the data using the accompanying ReadState.p program in 4GL code.

Defaulting Inputs with 4GL Programs

There will be times when one cannot simply pass over the data in a session variable. You may need to call on the system date or time, or make more advanced searches through the database combining tables with session variable parameters to pull up a default or do some action.

Here is a simple example of a program that can be used in conjunction with the plc() macro.

```
/* Today.p */
DEF OUTPUT PARAMETER cDate AS CHARACTER NO-UNDO.

ASSIGN cDate = STRING(TODAY, "99/99/99").
```

Example custom program for defaulting prompts

The prompt

```
@plc(Enter Start Date:,,Today.p)
```

will cause a prompt labeled Enter Start Date: defaulted to today’s date.

Using the @prg() macro

The @prg() macro also allows one to call out to programs to compute output. This allows more advanced data structure navigation and business logic to be entered into the report.

There are two arguments to the prg macro. The first is the name of the program to run. The second is the input you wish to feed that program. Currently there can only be one input

argument. If you require multiple input arguments, you should construct a string that you can slice and dice in your program.

You are allowed to include the `@fld()`, `@time()`, and `@date()` macros as an argument to the `@prg()` macro, as exemplified below:

```
@prg(Test.p,@fld(Judge.Initials))
```

You can also include strings to the macro:

```
@prg,(Test1.p,Hi There!)
```

Note that you should not use a comma to separate your arguments in a combined argument. This is because the comma is a meta-character to the macro. Perhaps a vertical slash would be best:

```
@prg(Test3.p,Arg1|Arg2)
```

Inside `Test3.p` you would use `Entry(1, YourInputParameter, "|")` to retrieve the first argument.

Javascript based totaling and subtotaling

(Note, with the advent of `@prg()` and other macros, many of these are about to become part of Denkh, which should be simpler to use. The javascript versions will remain for backwards compatibility, but should not be used in new reports.)

This is one of the more important uses for reports and there are tools available for the report designer to ease implementation of this.

If you wish to use them in letter reports, you should examine the code in `JSFunc.html` to insure it works with the HTML you plan on using.

For example, we wish to know the number of zip codes for cities starting with fl in the states:

City by Zip

Search City: fl

City	State	Zip	Count
Flat	AK	99584	1
Florence	AL	35630	8
Florence	AL	35632	
Flat Rock	AL	35966	
Florence	AL	35631	
Florence	AL	35634	
Floral	AL	36442	
Flomaton	AL	36441	
Florence	AL	35633	
Floral	AR	72534	2
Flippin	AR	72634	
Florence	AZ	85279	7
Flagstaff	AZ	86011	
Flagstaff	AZ	86003	
Flagstaff	AZ	86002	
Flagstaff	AZ	86001	
Flagstaff	AZ	86004	
Florence	AZ	85232	
Floriston	CA	96111	2
Flournoy	CA	96029	
Flagler	CO	80815	2
Florence	CO	81290	

Here is a record definition. We want to break by ZipCode.State in this example:

```
<script language="javascript">RptSubTotal("@fld(ZipCode.State)",3, 4, 0);</script>
<script language="javascript">TrkSubTotal("@fld(ZipCode.State)", 1, 0);</script>

<tr>
<td>@fld(ZipCode.City)</td><td>@fld(ZipCode.State)</td><td>@fld(ZipCode.ZipCode)</td>
</tr>
```

In the summary area, we also need to include this:

```
<script language="javascript">RptSubTotal("@fld(ZipCode.State)",3, 4, 0);</script>
```

We need to use two javascript functions that are automatically available with Denkh Reporter.

RptSubTotal(BreakBy, ColumnForSubTotal, NumberOfColumns, ID)

The purpose of this routine is to display the subtotal data within the data structures built into Denkh Reporter.

This routine takes the field used to break by. You should send in a database field that is ordered by index or a BY clause. Otherwise your data will be “scattered” and your subtotaling will be awkward.

The second argument is the column you wish the subtotal to appear in. Note that the column numbers start from 0 thru n, not 1 thru n. So if you want something to appear in column three, you should enter a two in this argument.

The next argument details the number of columns within the table. This however IS 1 thru n.

The last argument, ID, can be 0 – 4 to identify the subtotal from other subtotals.

You should always RptSubTotal() *before* TrkSubTotal() for a given ID. To get the last sub-total, you will need to call this in the Summary area.

TrkSubTotal(BreakBy, ValueToAdd, ID)

The purpose of this routine is to track the data within the data structures built into Denkh Reporter. To actually show the subtotal, you should use the RptSubTotal().

It takes three arguments; The Break By data field that is ordered in it’s retrieval, and the value to add to the subtotal.

The last argument, ID, can be 0 – 4 to identify the subtotal from other subtotals.

There are additional Javascript routines available:

SumSubTotal(ID)

This function provides a summary of the subtotals. It returns a table with a column of the values that are broken by and a column of the count of those entries.

The ID can be 0 – 4 to identify the subtotal from other subtotals.

CurSubTotal(ID)

This function returns the current subtotal for the iteration.

The ID can be 0 – 4 to identify the subtotal from other subtotals.

Counter(CounterID)

This routine provides a counter named CounterID. CounterID should be an integer number. It is a simple counter that iterates each time it is called.

RptCounter(CounterID, ID)

This routine displays the value of the Counter identified by CounterID.

Sigma (SigmaID, ValueToSum)

This routine allows the summation of multiple numbers of real and integer type. It is a good function to total by. SigmaID is an integer that identifies the sum. ValueToSum is a number to add into the sum.

RptSigma (SigmaID)

This routine reports the given sum.

Min, Max, Count and other mathematics

One can also define javascript based functions in the Page Header section. Then call out to these functions in the Records section. For example a max() function:

In the Page Header place:

```
<script language="javascript">
function max(a,b) {
  if (a > b) return a;
  return b;
}
</script>
```

In the records area:

```
<script language="javascript">
document.write(max(Avar, @fld(Invoice.Price)));
</script>
```

Planned Future Enhancements

- Report Wizards to aid in the construction of new reports
- Program modules which allow the user to add programming stored in the database and called by the @prg() macro. (The macro currently needs the module in the PROPATH of the transaction server.)
- Security which groups users into “groups” and then groups reports as accessible by that group.

If you have ideas for the software or critiques of this documentation, let me know!

If you have source code – send it to sauge@amduus.com. You will need to release all copyright and patent rights to the code to Amduus Information Works, Inc.

If you want to really help the project – SEND MONEY!!! Any amount you find you could save by using this program or the amount of value it might add to your organization would be nice:

Amduus Information Works, Inc.
1818 Briarwood
Flint, MI 48507