Denkh XML Reporter

Web Based Report Generation Software

Written By Scott Auge sauge@amduus.com
Amduus Information Works, Inc.
http://www.amduus.com

Table of Contents

License

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

What is it?

This software allows a programmer to call out to a web site running Progress' Webspeed or Amduus' Blue Diamond product, and request an XML formatted report from a given query.

Basic Software Requirements

- Progress RDBMS or Progress DataServer to another database (Progress supported OS)
- Progress Webspeed or Amduus Blue Diamond
- Software that can ask for an XML document via an HTTP request.
- 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 Designer Requirements

- Ability to code Progress Query strings
- Ability to create applications that can access the site via HTTP and parse the resulting XML document.

Advanced Report Designer Requirements

- Knowledge of E4GL programming
- Ability to create applications that can access the site via HTTP and parse the resulting XML document.

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.portal]
    srvrStartupParam=-p web/objects/web-disp.p -weblogerror -pf /db/amduus/parm/amduus.pf
     brokerLogFile=/tmp/portal.broker.log
    srvrLogFile=/tmp/portal.server.log
    uuid=192.168.254.254:80b00b4:e6e0749ab8:-68ab
    appserviceNameList=portal
    controllingNameServer=NS1
            PROPATH=/appl/Denkh/src
    workDir=/home/appl/opensrc/portal/amduus
```

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/report/rptgen/TestXMLRpt.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 /report/ portion refers to the cgi-bin program between the web server and the transaction server. 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 it can use. 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

It is really simple to create a report with this tool.

One basically crafts a Progress query, identifies the fields to bring back, and the maximum number of rows that will be recalled.

This information is packed up into an HTTP POST message and sent to the web server connected to a Progress database.

The XMLRpt.html program will return an XML document with a content-type of “text/xml” in the HTTP message.

One may need to unpack the XML document from the HTTP message if calling directly from another application. Often the programming tools available for generating the program has this ability.

The XML document is structured as so:

```
<ResultSet>
<Row>
<Column Name="[Name Of Field]" Order="[Column Number]">[Column
Data]</Column>
</Row>
</ResultSet>
```

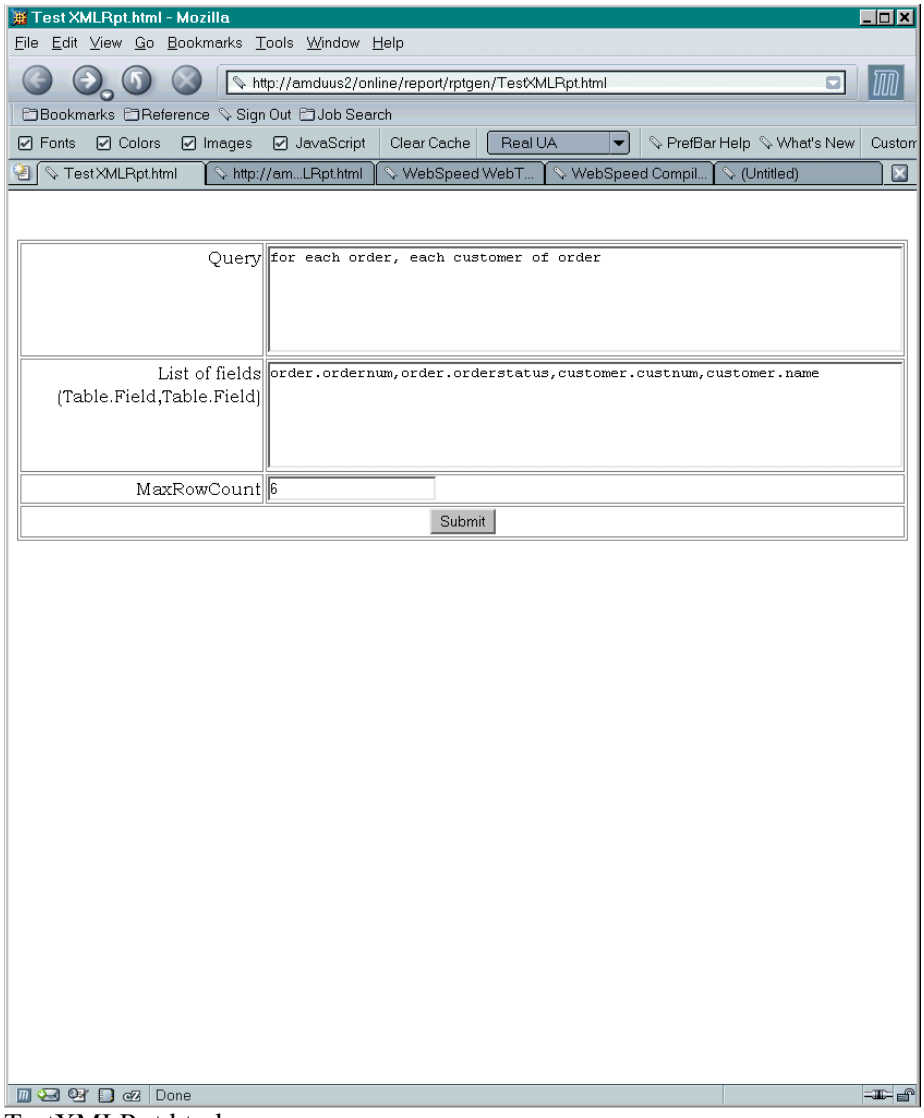
There will be a row for each result of the query.

The columns of the row are returned in the same order as they are presented in the FieldList NVP. Each column has an attribute for the name of the field (named by table.field) and the column number. This way, one can use the parser to refer to the column by name or by column number.

The data would be placed in the #PCDATA section of the tags (ie, between the <column> and </column> tags.

Basic Running of a Report

Two examples are given here. One example is via a web browser that can recognize an XML document and is good enough to present the document's tree.



TestXMLRpt.html.

The other example is calling for an XML document from the system from a common command line program on the Linux operating system.

A testing page

A simple program is available to test the system with. It is called

It allows the user to populate the three pieces of information XML Reporter needs to know in order to generate the XML document.

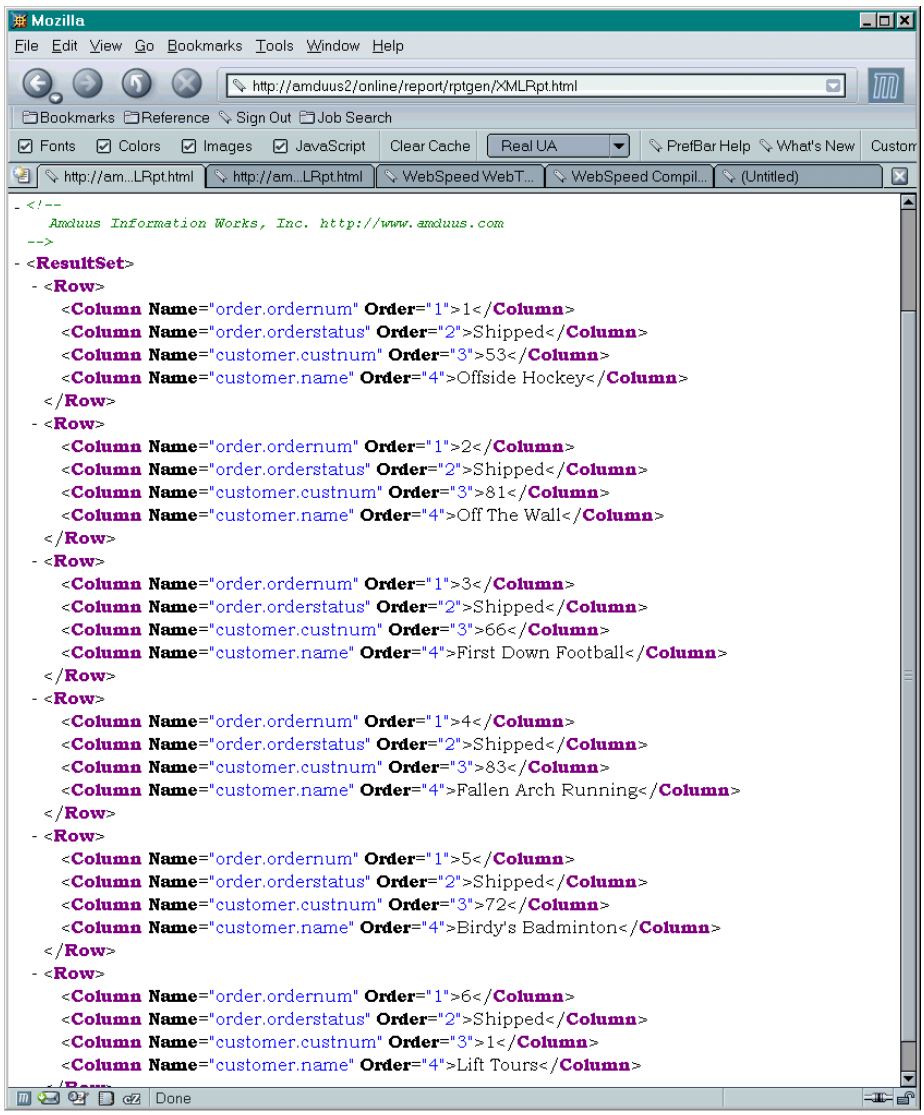
When done, it will POST the information to the program, which will generate the document.

Calling XMLRpt.html

The XMLRpt.html page returns an XML document in an HTTP message. You may need to pull the XML document out of the HTTP message. Often this can be done with tools used to create the application calling into the database via an HTTP message.

The Full URI for the is as follows.

XMLRpt.html?Query=[]&FieldList=[]&MaxCount=[]



Here is the text of the XML document….

```
<?xml version="1.0" ?>
<!-- Amduus Information Works, Inc. http://www.amduus.com -->
```

```
<ResultSet>
  <Row>
    <Column Name="order.ordernum" Order="1">1</Column>
    <Column Name="order.orderstatus" Order="2">Shipped</Column>
    <Column Name="customer.custnum" Order="3">53</Column>
    <Column Name="customer.name" Order="4">Offside Hockey</Column>
  </Row>
  <Row>
    <Column Name="order.ordernum" Order="1">2</Column>
    <Column Name="order.orderstatus" Order="2">Shipped</Column>
    <Column Name="customer.custnum" Order="3">81</Column>
    <Column Name="customer.name" Order="4">Off The Wall</Column>
  </Row>
  <Row>
    <Column Name="order.ordernum" Order="1">3</Column>
    <Column Name="order.orderstatus" Order="2">Shipped</Column>
    <Column Name="customer.custnum" Order="3">66</Column>
    <Column Name="customer.name" Order="4">First Down Football</Column>
  </Row>
  <Row>
    <Column Name="order.ordernum" Order="1">4</Column>
    <Column Name="order.orderstatus" Order="2">Shipped</Column>
    <Column Name="customer.custnum" Order="3">83</Column>
    <Column Name="customer.name" Order="4">Fallen Arch Running</Column>
  </Row>
  <Row>
    <Column Name="order.ordernum" Order="1">5</Column>
    <Column Name="order.orderstatus" Order="2">Shipped</Column>
    <Column Name="customer.custnum" Order="3">72</Column>
    <Column Name="customer.name" Order="4">Birdy's Badminton</Column>
  </Row>
  <Row>
    <Column Name="order.ordernum" Order="1">6</Column>
    <Column Name="order.orderstatus" Order="2">Shipped</Column>
    <Column Name="customer.custnum" Order="3">1</Column>
    <Column Name="customer.name" Order="4">Lift Tours</Column>
  </Row>
</ResultSet>
```

Another example

Another example is calling the program from the command line with a Linux program called wget. The program wget knows how to pull the document out of the HTTP message. Here is the command line to query the XML Reporter:

```
wget “http://localhost/online/report/rptgen/XMLRpt.html?Query=FOR+EACH+C
ustomer&MaxCount=5&FieldList=Customer.Name" –O /tmp/test.xml
```

wget places it’s results into a file of the name given after the –O option, in this case /tmp/test.xml.

By cat-ing the file, we can see it is an XML document:

```
[/tmp]$ cat test.xml
<?xml version="1.0" ?>
<!-- Amduus Information Works, Inc. http://www.amduus.com -->
    <ResultSet>
      <Row>
        <Column Name="Customer.Name" Order="1">Lift Tours</Column>
      </Row>
      <Row>
        <Column Name="Customer.Name" Order="1">Urpon Frisbee</Column>
      </Row>
      <Row>
        <Column Name="Customer.Name" Order="1">Hoops </Column>
      </Row>
      <Row>
        <Column Name="Customer.Name" Order="1">Go Fishing Ltd</Column>
      </Row>
      <Row>
        <Column Name="Customer.Name" Order="1">Match Point
Tennis</Column>
      </Row>
    </ResultSet>
```

When things go wrong

When a report blows up, a set 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

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)
```

An example error is:

```
<?xml version="1.0" ?>
<!-- Amduus Information Works, Inc. http://www.amduus.com -->
<ErrorList>
<Error>Prepare FAILED!
</Error>
<Error>
Open FAILED!
Date: 07/02/0315:06:53
Query: for some nontable
```

```
cBufferList:
for some nontable</Error>
</ErrorList>
```

Future Enhancements

Since Excel can read in XML documents, in the future the program may be adapted to create such documents.

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