DB2 Web Query Integration from PHP
Leveraging DB2 Web Query web services from IBM i applications

Jon Rush
IBM STG ISV Enablement
August 2009
Table of contents

Change history – optional ................................................................. 3
Abstract ..................................................................................................................... 4
Introduction ................................................................................................................. 4
Prerequisites ............................................................................................................. 4
The DB2 Web Query Web Service ................................................................. 5
Building the sample DB2 Web Query reports .................................................... 5
Create the Metadata for the QIWS Library......................................................... 6
Create the Balance Due bar chart report ......................................................... 9
Create the drill down report for the bar chart ............................................. 15
Create the Balance Due by State pie chart report ........................................ 22
Generate PHP classes for the web service .................................................... 30
The PHP samples .................................................................................................. 32
Sample 1 – Bar Chart drill down ................................................................. 32
DB2WEBQUERY5A.PHP ..................................................................................... 33
TheWebQueryRunFex section ........................................................................ 35
This web service also requires the DB2 Web Query domain, the folder and the name of
the report. You can retrieve this information by looking at the properties of the report
created in the The DB2 Web Query Web Service ............................................. 36
The WebQueryLink section ........................................................................ 39
DB2WEBQUERYDRILL.PHP ............................................................................ 42
WebQuery.php ..................................................................................................... 44
WenQueryMisc.php .............................................................................................. 48
HTML files ........................................................................................................ 49
Running the first bar chart sample ................................................................. 49
Sample 2 – Pie Chart ............................................................................................. 52
Running the second pie chart sample ............................................................. 53
Appendix A. A Java framework for DB2 Web Query SDK ............................... 54
Summary ................................................................................................................. 55
Resources ............................................................................................................. 56
About the author .................................................................................................. 56
Trademarks and special notices ........................................................................ 57
### Change history – optional

Delete this section if this is the last version of the paper.

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Editor</th>
<th>Editing description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>2009-08-06</td>
<td>Jon Rush</td>
<td>Original version</td>
</tr>
<tr>
<td>2.0</td>
<td>2009-08-27</td>
<td>Jon Rush</td>
<td>Kent Milligan updates</td>
</tr>
<tr>
<td>3.0</td>
<td>2009-09-04</td>
<td>Jon Rush</td>
<td>Gen Cobb updates</td>
</tr>
<tr>
<td>4.0</td>
<td>2009-09-10</td>
<td>Jon Rush</td>
<td>Corrections/Additions</td>
</tr>
</tbody>
</table>
Abstract

This white paper explores the use of the DB2 Web Query for i Software Development Kit (SDK) with PHP applications running on IBM i. Included in this paper is a reusable PHP application that may be used as a starting point for PHP application providers to integrate DB2 Web Query web services into their web applications.

Introduction

The IBM DB2 Web Query for i product (5722QU2) is a powerful tool for creating queries and reports in a Web-based environment. It offers a set of modernized tools for a more robust, extensible and productive reporting solution than the popular Query for System i (also known as Query/400) tool. DB2 Web Query for i preserves investments in the reports developed with Query/400 by offering a choice of importing definitions into the new technology or continuing to run existing Query/400 reports as is. But it also offers significant productivity and performance enhancements by leveraging the latest in DB2 for i query optimization technology.

The DB2 Web Query Software Development Kit (SDK) was recently released as a productivity enhancement to the DB2 Web Query for i product. This toolkit exposes the reports, graphs and queries to applications as open web service programming interfaces. Applications written in Visual Basic/.NET, Java, RPG, and C can leverage the web service to integrate the dynamic reports into their frameworks. Integration of graphics and dynamic reports into PHP web applications is growing and other frameworks do exist in this space. The Business Intelligence Reporting Tool (BIRT) is one that is similar in function to the DB2 Web Query SDK. BIRT has an eclipse plugin to build reports from relational databases, and then runs these reports in an Java runtime engine that runs in most industry application servers. PHP applications use a product called Javabridge to invoke the BIRT runtime engine in the application server. Jasper is another open source framework for building these types of applications and is built to run in open applications servers leveraging XML and web services as integration points.

The purpose of this technical paper is to give the reader a concrete example of how to use these web services from System i based PHP web applications. The popularity of the Zend Core and Zend Platform PHP products on System i necessitates the need for examples of how to enhance PHP/System i applications with dynamic web query functionality.

Prerequisites

The prerequisite setup for this paper and sample code is an IBM i OS V5R4 or higher system with DB2 Web Query installed with the Web Services SDK along with Zend Core for i5/OS.

- DB2 Web Query product 5733-QU2
- DB2 Web Query Software Development Kit 5733-QU4
- Zend Core for i5/OS or Zend Platform for i5/OS
- DB2 Web Query Run Time User Enablement 5733QU2 option 4
The DB2 Web Query Web Service

The DB2 Web Query SDK is a set of web services exposed to allow programmatic use of web query reports and report broker functions from .NET, Java, RPG or PHP applications. The WebFocus Web Query Web Services guide referenced in the resource section at the end of this paper documents the services. This paper uses several of these functions to perform drill down graphic report capability in a PHP application. The key web query services used by the PHP samples are:

- **WebQueryLogOn** – This will authenticate the application against the DB2 Web Query environment security mechanism. The successful execution of this service and the returned LogInfo object contains information that is used by all subsequent service invocations after this to authenticate the user executing the service functions.

- **WebQueryRunFex** – This service will run a Web Query report. The LogInfo credential/authentication object is required along with the FexInfo object which identifies the report to be run, along with any parameters that are to be used by the query.

- **WebQueryLink** – This service will run links returned in the output of the WebQueryRunFex service. In the PHP samples provided this is used to get the image file built dynamically by running the web query report with the WebQueryRunFex service.

- **WebQueryDrill** – This service will run drill down links returned in the WebQueryRunFex result page. In the PHP samples here this will be used when a clickable section of a query graphic image map is clicked on in the browser window.

Building the sample DB2 Web Query reports

It is assumed that the reader has access to a system with DB2 Web Query and a Zend PHP server. It is also assumed that DB2 Web Query skills are available since this section is a basic process for setting up DB2 Web Query reports using the Graph assistant and Report assistant features of DB2 Web Query. In this section there are many screen captures to guide you through building the reports used by the sample code. The Resources section at the end of this document contains several links to documentation and IBM Redbooks where DB2 Web Query education and skills can be attained.
Create the Metadata for the QIWS Library

1. Open your web browser and log on to your DB2 Web Query for i environment. Create a new Report Folder named qcustcdt.
2. Once the qcustcdt folder has been created, right mouse click on it and select “Metadata”.

3. On this screen right-click on the “LOCAL adapter under DB2 cli, select “Create Synonym”. It is recommended to use the DB2 cli adapter when building reports that access SQL tables or single-member physical files. This is because the DB2 cli adapter will generate SQL to perform the database access request. The QCUSTCDT file used in our example is a single-member physical file containing balance and credit information for customers.
4. Select Tables and enter QIWS in the Library field so we can create metadata on tables in the QIWS library. Click the Next button to continue.

5. Select the QCUSTCDT table as shown in this figure and click Create synonym. You will then see the confirmation page with the status of the synonym create for qcustcdt. You may now close this window.
Create the Balance Due bar chart report

Now that we have the metadata defined for the QCUSTCDT table, we will use the Graph Assistant to create a new report. This report will be used in the first web service example. The query takes one parameter (balance due) to return customers with balances due greater than the inputted value.

The next figure shows what the PHP application will look like. The leftmost page is the web form where users can input the value for balance due that the query will use as its selection criteria parameter. Using that input value the web query web service will be invoked to return a bar chart as shown in the rightmost page. The bars in the chart will be “clickable” to show drill down information as shown in the bottom frame in the bar chart page.

![Figure 1 Bar chart PHP application](image)

The next series of screens guides the reader through creation on the web query reports. Log on to your DB2 Web Query application.
Create the bar chart report

1. Right click on the qcustcdt report folder and select Graph Assistant.

2. Select the QCUSTCDT database description created in the prior section and click OK.
3. Next decide what type of chart to use. We will select a Clustered Bar Chart.

4. On the Field Selection tab select LSTNAM which contains the customer's last name from the Available fields table and click the icon above the “Field value plotted on X axis” field. If you are using Microsoft Internet Explorer, you may alternatively “drag and drop” the LSTNAM field into the “Field value plotted on X axis” field.
5. Select BALDUE and click the icon on the “Field values plotted on Y axis”. (Again if you are using Microsoft Internet Explorer, you may alternatively “drag and drop” the BALDUE field into the “Field value plotted on Y axis” field)

6. On the Selection criteria tab, select BALDUE in the left pane and click the icon. Pull down to select GREATER THAN, then click on <Select values> in the large pane at the right to complete the where selection criteria. This criterion will be used to constrain the query to return only LSTNAM rows where BALDUE is greater than a user inputted value.
7. Select Parameter and enter baldue which is the input balance due parameter, then click OK which will return you to the Selection criteria panel.

8. With the Selection criteria completed, it's time to save the graph definition. Click the pull down arrow beside the Save menu item.
9. Select the "Save As" task and save the report as "Balance Due Bar Chart – no prompt". NOTE: Make sure the "prompt for parameters" is Unchecked as shown. We uncheck this since the parameters will be passed in programmatically via the web service and not from a separate dialog prompt box. Click OK and then Click Quit button on the main Graph Assistant window.
Create the drill down report for the bar chart

1. Next we create another report named "qcustcdt report - no prompt". This report will be invoked when the "Balance Due Bar Chart – no prompt" is presented and contains clickable bars that enable drill down into a customer's detail data. Select your folder and right click, selecting Report Assistant.

2. Select the QCUSTCDT database description created in the prior section and click OK.
3. On the Field selection tab select those fields shown and drag them into the Sort by pane. (or use the icon)

4. As shown in this figure, on the Selection criteria tab pick LSTNAM and press the icon then click on the <Select values> link in the large text area on the right.
5. Select the Parameter radio button and enter lstnam in the text box and click on the icon. Then click OK. This is the parameter that will be passed from the original query to drill down and report the customer’s detailed data.

6. Back on the Selection criteria panel, select the pull down arrow beside the Save menu item. Select Save As and save as shown in the next figure.
7. Save this report as shown. Click OK and then Click Quit button on the main Report Assistant window.

Now we have to re open the "Balance Due Bar Chart – no prompt" report to add linkage to the drill down report (qcustcdt report – no prompt) we just created in the above section.

1. Re open the bar chart with a right click on the Balance Due Bar Chart – no prompt report and select Open.
2. Click on the Field selection tab. Select BALDUE field on the right side in the Sum panel and then open Field Options for Baldue at the bottom. Click the Drill down button.

3. Click Execute procedure radio button. Once this radio button is selected, DB2 Web Query will populate the first text box with your report folders. In this text box, find the "qcustcdt report – no prompt" report (procedure) and select it with a left-click. The left-click populates the Execute procedure box with the DB2 Web Query path for the selected report. Next, click the Add button so that the required input parameter can be passed to the report.
4. In the parameter name text field enter lstnam (this must be the same as the parameter specified in step 4 of the Create the drill down report for the bar chart section – it is case sensitive so make sure that the names match exactly) and the Parameter value, Field radio button text field should be LSTNAM. Click OK when complete. This will pass the LSTNAM column data to the drill down report to get customer detail data. Back on the drill down dialog window, click OK.
5. On the Graph Assistant window, click Save menu option. A window stating the report is saved pops up, click OK and then press Quit to end the Graph Assistant.
**Create the Balance Due by State pie chart report**

Now we will create the reports for the second web services example in this paper. This report is a drill down pie chart of balances due by state. By clicking on the state, the application will drill down and show the customer accounts in the selected state. The next figure shows the running PHP application.

![DB2 Web Query Drill Down](image_url)

**Figure 2 Pie chart example**
1. On your report folder in your domain, right click and start Graph Assistant.

2. Select the QCUSTCDT database description.
3. Select PIE for type of chart with BASIC for the Graph Style.

4. On the Field Selection tab complete the panel as shown. STATE column for the X axis and BALDUE for the Y axis as shown. Save this report as “Balance Due By State”. The Y axis value, BALDUE, will contain the sum of balance due columns in each X axis value (STATE).
5. Next we create another report named “State drill down”. Select your folder and right click, selecting Report Assistant.

6. Select the QCUSTCDT database description.
7. Select the fields shown in the left text area and press the icon next to the Sort by text area.

8. Next go to the Selection criteria tab. Select the STATE field and click the icon above the right text area. Click on the <Select values> link.
9. Select the Parameter radio button and type state into the field, then click the icon. Click OK.

10. Select the pull down arrow next to the Save menu item.
11. Save the report as “State drill down”. Click OK.

Now we have to re open the “Balance Due By State report” to link it with the drill down report (State drill down) just created in the above section.

1. Right click on the Balance Due By State report.
2. On the Field selection tab, select BALDUE in the “Field values plotted on Y axis” text area. Click on the Show – Field options – BALDUE. You will then see the options for BALDUE. Click on Drill down.

3. Click on “Execute Procedure (FOCEXEC)”. In the top text box, expand your domain and folder (qcustcdt) and pull down until you can select the State drill down we just created. Then click the “Add” button.
Now that the up front work is done of defining the reports to be used and invoked from our PHP samples, we can move on to seeing how to generate the PHP classes needed to invoke the DB2 Web Query web services from our PHP applications.

### Generate PHP classes for the web service

We have used the WSDL2PHP utility from SourceForge: [http://sourceforge.net/projects/wsl2php/](http://sourceforge.net/projects/wsl2php/) as the utility that will programatically parse the DB2 Web Query WSDL file and create PHP classes for interacting with the web service. The generated PHP include file (WebQuery.php) is included in the zip file with the samples, but you can follow the next steps to generate yourself.

1. Extract the wsdl2php.php file into the /usr/local/Zend/Core/bin IFS directory on your PHP server running on IBM i.

2. Get the DB2 Web Query web service WSDL file from your DB2 Web Query host system. This can be done by using your browser to open the following URL and save the resulting page into the document root of your System i PHP server. For example /www/zendcore/htdocs.

   http://target_machine[:port]/webquery/uddi/WebQuery.jsp?wsdl
3. Run the wsdl2php.php utility against the WSDL file saved in the preceding step. This will create a WebQuery.php file in the /usr/local/Zend/Core/bin IFS directory. Move this file to your PHP server document root. For example /www/zendcore/htdocs.

   php wsdl2php.php /www/zendcore/htdocs/WebFocusFexUddiWsdl.xml

4. The WSDL for the web query services has two "dataset" classes defined. One class named "DataSet" and the other named "dataset" - these two classes are identical other than the class name. PHP class names used with the new command are case-insensitive. We have to edit this file to comment out or delete the duplicate class definition. See Figure 3 WebQuery.php changes.
Delete the following class definition for “dataset” or Comment out these lines by putting a “//” at the
beginning of each as shown:

```php
//class dataset { change to wqdataset
// public $schema; // <anyXML>
  // public $any; // <anyXML>
//}
```

**Figure 3 WebQuery.php changes**

**The PHP samples**

There are two samples included. The first is a simple bar chart returned based on the amount of balance
due on customer accounts. You can then click on the bars to drill into the customer information. It
illustrates the main steps in using the DB2 Web Query web services to invoke a report and execute an
associated drill-down reports.

**Sample 1 – Bar Chart drill down**

This first PHP application consists of:

- **db2webquery5a.php**
  This is the main PHP script that invokes the WebQueryRunFex web services to run the reports
  created earlier in the paper. It modifies the returned HTML to change the HREFs in the image
  map to invoke the db2webquerydrill PHP script when the hotspots are clicked on.

- **db2webquerydrill.php**
  This is the PHP script that invokes the WebQueryDrill web service to execute the drill down
  query.

- **WebQuery.php**
  This is the generated classes created by WSDL2PHP for invoking the DB2 Web Query web
  services. You can use this file if you do not want to generate it as in the Generate PHP classes
  for the web service section.

- **WebQueryMisc.php**
  This is an include file with some predefined variables and classes including the host name and
  port of the PHP server and the IBM i user id and password.

- **db2webqueryframes.html**
  This is the HTML page that identifies the frameset and frames for the sample application.

- **db2webquerydrillstart2.html**
  This is the HTML page with the FORM for entering the balance due variable that is used to
determine selection criteria for the query.
DB2WEBQUERY5A.PHP

WebQueryLogOn section

//
// DB2 Web Query with input parameters - Drill down
//
include_once 'WebQuery.php';
include_once 'WebQueryMisc.php';
header("Cache-Control: no-store, no-cache, must-revalidate");
header("Cache-Control: post-check=0, pre-check=0", false);
session_start();
$WQSoapClient = new WebQuery();
$WQLogonReq = new WebQueryLogOn();
$WQLogonReq->userid='';
$WQLogonReq->pass='';
// mreuiday and mrepass from the WebQueryMisc.php include file
$WQLogonReq->mreuiday=$WQuserinfo->user;
$WQLogonReq->mrepass=$WQuserinfo->pw;
$WQLogResult = $WQSoapClient->WebQueryLogOn($WQLogonReq);
//Check to see if logon successful
if ($WQLogResult->WebQueryLogOnResult->status != true) {
    echo('<h1>WebQueryLogOn FAILED! Check USER and PASSWORD</h1>');
    die();
}
//PHP session will hold the Web Query Logon credentials to be reused in
//subsequent web service invocations.
$_SESSION['wqloginfo']=$WQLogResult->WebQueryLogOnResult;

Figure 4 Web Query Logon
In Figure 4 Web Query Logon the key areas are:

1. The header functions insert Cache-Control headers into the HTML page being sent back to the web browser from this PHP script to insure that the browser will not cache images and pages in its local cache so that the application will always go back to our PHP and DB2 Web Query server for results.

2. The PHP session handling is initiated with the session_start() function that will create a new session id for this web application.

3. The next several lines create a new WebQuery object named $WQSoapClient and a WebQueryLogOn object named $WQLogonReq. The $WQSoapClient object is the object that has functions defined to invoke the various web query web services. The $WQLogonReq is the object that has the data elements needed for a WebQueryLogOn web service request. Required fields in that object are $WQLogonReq->mreuid and $WQLogonReq->mrepass which we get from the WebQueryMisc.php include file. This is a valid IBM i userid and password on your DB2 Web Query server.

4. The WebQueryLogOn web service is invoked with the $WQLogResult = $WQSoapClient->WebQueryLogOn($WQLogonReq); statement. The logon credentials from a successful logon are stored into the php session with the $_SESSION['wqloginfo']=$WQLogResult->WebQueryLogOnResult; statement. In this sample error checking from the WebQueryLogOn is done by checking if $WQLogResult->WebQueryLogOnResult->status != true to make sure the logon completed successfully.
**The WebQueryRunFex section**

```php
//Array of ValuesArrayEntries for passing values to the db2 web query

$WQValuesArrayEntry = array(new ValuesArrayEntry());

$WQValuesArrayEntry[0]->name="baldue";

$WQValuesArrayEntry[0]->val=$_GET[baldue];

$WQWebQueryRunFex = new WebQueryRunFex();

$WQWebQueryRunFex->IBIWS_cookie = new LogOnInfo();

$WQWebQueryRunFex->IBIWS_cookie = $_SESSION['wqloginfo'];

$WQWebQueryRunFex->IBIWS_fexinfo = new FexInfo();

$WQWebQueryRunFex->IBIWS_fexinfo->MREdomain='l0lkv6tr/l0lkv6tr.htm';

$WQWebQueryRunFex->IBIWS_fexinfo->MREfolder='#qcustcdtlmvl';

$WQWebQueryRunFex->IBIWS_fexinfo->name='app/balance_due_bar_chart_no_prompt.fex';

$WQWebQueryRunFex->IBIWS_fexinfo->IBIWS_arrayvalues=$WQValuesArrayEntry;

$WQWebQueryRunFexResult = $WQSoapClient->WebQueryRunFex($WQWebQueryRunFex);
```

**Figure 5 WebQueryRunFex**

1. In this section of the PHP code, we are preparing to invoke the WebQueryRunFex web service to run the non-promptable report we set up in the Create the Balance Due bar chart report section. The first thing we do is to create an array of ValuesArrayEntry types as defined in the WebQuery.php include file (that was generated from the WSDL file). The structure of this object is shown in Figure 6 Values Array Structure below. As you can see we set an entry with name="baldue" and set its' value to the value being passed into this script from an input form on a preceding web page in the $_GET variable named baldue.

```php
class ValuesArrayEntry {
  public $name; // string
  public $val; // string
  public $format; // string
  public $prompt; // string
  public $defaultVal; // string
  public $multi; // boolean
  public $quote; // boolean
}```
Figure 6 Values Array Structure

2. The next few lines create a new WebQueryRunFex object that will contain the logon credentials we saved in the PHP session from the WebQueryLogOn invocation in Figure 4 Web Query Logon.

This web service also requires the DB2 Web Query domain, the folder and the name of the report. You can retrieve this information by looking at the properties of the report created in the The DB2 Web Query Web Service

The DB2 Web Query SDK is a set of web services exposed to allow programmatic use of web query reports and report broker functions from .NET, Java, RPG or PHP applications. The WebFocus Web Query Web Services guide referenced in the resource section at the end of this paper documents the services. This paper uses several of these functions to perform drill down graphic report capability in a PHP application. The key web query services used by the PHP samples are:

- WebQueryLogOn – This will authenticate the application against the DB2 Web Query environment security mechanism. The successful execution of this service and the returned LogInfo object contains information that is used by all subsequent service invocations after this to authenticate the user executing the service functions.

- WebQueryRunFex – This service will run a Web Query report. The LogInfo credential/authentication object is required along with the FexInfo object which identifies the report to be run, along with any parameters that are to be used by the query.

- WebQueryLink – This service will run links returned in the output of the WebQueryRunFex service. In the PHP samples provided this is used to get the image file built dynamically by running the web query report with the WebQueryRunFex service.

- WebQueryDrill – This service will run drill down links returned in the WebQueryRunFex result page. In the PHP samples here this will be used when a clickable section of a query graphic image map is clicked on in the browser window.

3. Building the sample DB2 Web Query reports section. The Figure 7 Select properties and Figure 8 Properties dialog show the properties for the balance due bar chart report that is needed by the
web service as input parameters. Specifically the IBIWS_fexinfo->MREdomain, IBIWS_fexinfo->MREfolder, and IBIWS_fexinfo->name as shown in the code above.

Figure 7 Select properties
4. We then set the WebQueryRunFex service input values array to our $WQValuesArrayEntry which contains the value for baldue that was passed in from the web form. This value is used by the Balance Due Bar Chart report we built in the Create the Balance Due bar chart report section.

5. The last step in this section is to execute the WebQueryRunFex. The $WQWebQueryRunFexResult = $WQSoapClient->WebQueryRunFex($WQWebQueryRunFex); statement invokes the web service and places the result in the $WQWebQueryRunFexResult object that is used by the next section of code. This result contains three key things that will be referenced to complete the page return to the browser.

- An object that contains an array of links that include the image file for the clickable image map for drill down.
- In that same link array, the links for the drill down db2 web query reports for each clickable item.
- The HTML page that contains the image map and HREFs for the drill downs.
The section of code shown in Figure 9 WebQueryLink invocation is where the result of the WebQueryRunFex service is handled. The result structure contains an array of links returned that contains, in this example, a link for the PNG (Portable Network Graphics) image that was created by running the Balance Due Bar Chart report. The while loop goes through the links array and extracts the link for the PNG file link.

The next section of code in Figure 9, instantiates a new WebQueryLink structure. WebQueryLink is the web service to invoke which will execute the links returned from running a report. In this example, when the WebQueryRunFex is invoked it will return several links to the PHP application. These links will include http links to graphic image files in .jpg, .gif, or .png format, and http links to the URL to invoke drill down reports. As with all the invocations to the web service, it uses the LogInfo object store in the PHP session. The other parameter that it requires is the link that you want to run. The first section of code in Figure 9 WebQueryLink invocation, loops through the links returned by running the WebQueryRunFex in the previous section. It will stop when it finds the "png" link. It then constructs the WebQueryLink parameter object and invokes the link and saves the png (image) file into a .png file in the document root of the Zend PHP server.

```php
$LinkArrayEntry = new LinkArrayEntry();
$LinkArrayEntry = $WQWebQueryRunFexResult->WebQueryRunFexResult->links;
// Get the image file from the WebQueryRunFex.
$i = 0;
while (($LinkArrayEntry->LinkArrayEntry[$i]->type) != "png") {
    $i = $i + 1;
}
$WQWebQueryLink = new WebQueryLink();
$WQWebQueryLink->IBIWS_cookie = new LogOnInfo();
$WQWebQueryLink->IBIWS_link = new LinkArrayEntry();
$WQWebQueryLink->IBIWS_cookie = $_SESSION['wqloginfo'];
$WQWebQueryLink->IBIWS_link = $LinkArrayEntry->LinkArrayEntry[$i];
$WQWebQueryLinkResult = $WQSoapClient->WebQueryLink($WQWebQueryLink);
//save the binary data (PNG) into a file in the ZEND PHP server docroot
$theFile = fopen("fileOne.png", "w");
fputs($theFile, $WQWebQueryLinkResult->WebQueryLinkResult->binaryData);
fclose($theFile);
```
Now the application will again loop through the links returned to create an array of drill down links. For illustrative purposes this second loop was created, but could have been incorporated into the loop above that was used to find the image file.

```php
//Now loop through the links to find fexdrill links and store in array.

//
$numLinks = count($LinkArrayEntry->LinkArrayEntry);
$j = 0;

for ($i = 0; $i <= ($numLinks-1); $i++) {
    $linkType = $LinkArrayEntry->LinkArrayEntry[$i]->type;
    if ($linkType == "fexdrill") {
        $fexdrill[$j]=$LinkArrayEntry->LinkArrayEntry[$i]->link;
        $j++;
    }
}
```

Figure 10 Getting drill down links

The final step is to modify the HTML output returned from the WebQueryRunFex. The next figure shows the image map returned from the web query web services. It shows one of the AREA tags for the drill down along with the HREF to be invoked when clicked. The HREF you see below is one of the drill down links return in the array of links returned in the WebQueryRunFex result object. In Figure 10 we extracted these into a PHP array named $fexdrill.

```html
<MAP NAME=xvmxlyen0>
<AREA SHAPE=POLYGON COORDS="90,276,160,276,160,247,90,247"
NAME="BALDUE Abraham 500.00"
TITLE="BALDUE Abraham 500.00"
HREF="/webservice?IBIF_webapp=%2Fwebquery&IBIC_server=EDASERVE&IBIWF_msgviewer=OFF&IBIMR_dri ll=X,l0lkv6tr%2Fl0lkv6tr.htm&IBIF_ex=l0lkv6tr%2Fapp%2Fcustc0d_report_no_prompt.fex&CLICKED_ON=UNS ET%20PARAMETER&lstnam=Abraham&"
ALT="BALDUE Abraham 500.00">
```
We are going to use the PHP DOM (document object model) classes to modify this HTML. In the following code section all instances in the HTML output string where: HREF="/webservice?.. as shown in Figure 11 to -> HREF="http://<your php server host:port>/db2webquerydrill.php?wqsession=<php session id>&fexlink=<with the original HREF link from above>. So for example our new HTML page would have an HREF link that now is:

```
HREF=http://em15a.rchland.ibm.com:89/db2webquerydrill.php?wqsession=t14mkm8bhr4t98b4t4av17c16&amp;fexlink=%2Fwebserver%3FIBIF_webapp%3D%26IBIC_server%3DEDASERVE%26IBIF_msgviewer%3DUNSET%26IBIMR_drill%3Dl0lkv6tr%252Fl0lkv6tr.htm%26IBIF_ex%3Dl0lkv6tr%252Fapp%252Fqcustcdt_report_no_prompt.fex%26CLICKED_ON%3DUNSET%2520PARAMETER%26lstnam=Abraham%2
```

This new HREF link now invokes a new utility php script named "db2webquerydrill.php" that resides in the document root of the Zend PHP server. We also include the PHP session id as a parameter to this utility. We will discuss this utility in the next section.

We have one final thing to modify in the HTML document. We need to change the IMG tag to point to the PNG file we saved on the Zend Server in Figure 9.

```
<IMG SRC="/webservice?&PG_Func=GETBINARY&PG_File=qjqemqjp.png"
```

Gets transformed to this:

```
<img src=http://em15a.rchland.ibm.com:89/fileOne.png
```

The complete code section for this is shown in the next figure.
<?php
//
// DB2 Web Query with input parameters - Drill down
  //
include_once 'WebQuery.php';

Figure 12 Modify HTML and send page to browser

**DB2WEBQUERYDRILL.PHP**

This PHP script is invoked when the clickable area of an image is select and clicked on. The invocation has two parameters. The first is the PHP session ID that has the all important LogOnInfo object that is the authentication credentials that DB2 Web Query needs to process requests. The second parameter is the link to be passed to the WebQueryDrill web service to initiate the drill down report. We execute the drill down and then echo the HTML output returned from that web service.
header("Cache-Control: no-store, no-cache, must-revalidate");
header('Pragma: no-cache');
header('Expires: -1' );
//header("Cache-Control: post-check=0, pre-check=0", false);
session_id($_GET[wqsession]);
session_start();

$WQSoapClient = new WebQuery();
/*
$WQLogonReq = new WebQueryLogOn();
$WQLogonReq->userid='';
$WQLogonReq->pass='';
$WQLogonReq->mreuid=$_SESSION['uid'];
$WQLogonReq->mrepass=$_SESSION['pwd'];
$WQLogResult = $WQSoapClient->WebQueryLogOn($WQLogonReq);
*/
$WQWebQueryDrill = new WebQueryDrill();
$WQWebQueryDrill->IBIWS_cookie = new LogOnInfo();
$WQWebQueryDrill->drill = $_GET[fexlink];
$WQWebQueryDrill->IBIWS_cookie = $_SESSION['wqloginfo'];
//$WQWebQueryDrill->IBIWS_cookie = $WQLogResult->WebQueryLogOnResult;
$WQWebQueryDrillResult = $WQSoapClient->WebQueryDrill($WQWebQueryDrill);

echo ("<center">.$WQWebQueryDrillResult->WebQueryDrillResult->output);
?>

Figure 13 db2webquerydrill utility php script
**WebQuery.php**

This is the PHP that was generated by the WSDL2PHP utility. It contains the classes and methods to use for invoking the web services. This is not the complete file, but shows you a few of the classes used. For example the LogOnInfo class is a key class used by the web service and this application. It is returned from the call to the WebQueryLogOn service and is stored in the PHP session in our sample code. Each subsequent call to the web query web services will reuse this object as authentication for the DB2 Web Query server. The WebQueryLogOn class is also shown along with the WebQueryLogOnResponse class. Note the response class contains an instance of the LogOnInfo class. Also shown in the following figure is the WebQuery class which extends SoapClient. It contains all the functions used to invoke the web services. There is a WebQueryLogOn function in the WebQuery class that executes a soapCall to the WebQueryLogOn service on our DB2 Web Query server and passes in a WebQueryLogOn class as the parameter to the service call.

```php
<?php
...

class LogOnInfo {
    public $time; // long
    public $status; // boolean
    public $MRcookie; // string
    public $WFcookie; // string
    public $WFviewer; // string
    public $userid; // string
    public $pass; // string
    public $mreuid; // string
    public $mrepass; // string
    public $cserver; // string
    public $dosignon; // boolean
    public $othercookies; // ArrayOfCookiesArrayEntry
}
...

class WebQueryLogOn {
    public $userid; // string
    public $pass; // string
```
public $mreuid; // string
public $mrepass; // string
}

class WebQueryLogOnResponse {
    public $WebQueryLogOnResult; // LogOnInfo
}
...

/**
 * WebQuery class
 *
 * created at 20 Jul 2009 14:35:51 BST
 *
 * @author {author}
 * @copyright {copyright}
 * @package {package}
 */

class WebQuery extends SoapClient {

    private static $classmap = array(
        'edanodeObject' => 'edanodeObject',
        'MultiOutputFormat' => 'MultiOutputFormat',
        'MultiOutputType' => 'MultiOutputType',
        'DataSet' => 'DataSet',
        'InputData' => 'InputData',
        'schema' => 'schema',
        'xml' => 'xml',
        'dataset' => 'dataset',
        'WebQueryReturn' => 'WebQueryReturn',
        'LinkArrayEntry' => 'LinkArrayEntry',
    )
}
'LinkType' => 'LinkType',
'StringArray' => 'StringArray',
'CookiesArrayEntry' => 'CookiesArrayEntry',
'ValuesArrayEntry' => 'ValuesArrayEntry',
'FexInfo' => 'FexInfo',
'LogOnInfo' => 'LogOnInfo',
'MasterInfo' => 'MasterInfo',
'ArrayMasterInfoEntry' => 'ArrayMasterInfoEntry',
'MasterInfoEntry' => 'MasterInfoEntry',
'DBInfo' => 'DBInfo',
'ArrayDBInfoEntry' => 'ArrayDBInfoEntry',
'DBInfoEntry' => 'DBInfoEntry',
'ServerInfo' => 'ServerInfo',
'ArrayserversArray' => 'ArrayserversArray',
'MREReturn' => 'MREReturn',
'WSDLInfoStruct' => 'WSDLInfoStruct',
'WSDLInfoEntry' => 'WSDLInfoEntry',
'WebQueryDrill' => 'WebQueryDrill',
'WebQueryDrillResponse' => 'WebQueryDrillResponse',
'WebQueryRunFex' => 'WebQueryRunFex',
'WebQueryRunFexResponse' => 'WebQueryRunFexResponse',
'WebQueryFexReflection' => 'WebQueryFexReflection',
'WebQueryFexReflectionResponse' => 'WebQueryFexReflectionResponse',
'WebQueryLink' => 'WebQueryLink',
'WebQueryLinkResponse' => 'WebQueryLinkResponse',
'WebQueryFieldValues' => 'WebQueryFieldValues',
'WebQueryFieldValuesResponse' => 'WebQueryFieldValuesResponse',
'WebQueryLogOn' => 'WebQueryLogOn',
'WebQueryLogOnResponse' => 'WebQueryLogOnResponse'}
'MREGetUserDomains' => 'MREGetUserDomains',
'MREGetUserDomainsResponse' => 'MREGetUserDomainsResponse',
'MREOpenDomain' => 'MREOpenDomain',
'MREOpenDomainResponse' => 'MREOpenDomainResponse',

public function WebQuery($wsdl = "webqry.wsdl", $options = array()) {
    foreach(self::$classmap as $key => $value) {
        if(!isset($options['classmap'][$key])) {
            $options['classmap'][$key] = $value;
        }
    }
    parent::__construct($wsdl, $options);
}
...
/**
 * @param WebQueryLogOn $parameters
 * @return WebQueryLogOnResponse
 */
public function WebQueryLogOn(WebQueryLogOn $parameters) {
    return $this->__soapCall('WebQueryLogOn', array($parameters), array(
        'uri' => 'http://informationbuilders.com/',
        'soapaction' => ''
    ));
}...
Figure 14 WSDL2PHP generated classes

WenQueryMisc.php

This file contains some application specific constants and classes. Note the values in **BOLD/Italicized** font that must be changed for your environment.

```php
<?php

class db2webqrynode {
    public $host; // string
    public $port; // string
}

class db2webqryuser {
    public $user; // string
    public $pw; // string
}

$WQhostinfo = new db2webqrynode();
// PUT YOUR ZEND HOST NAME and LISTENING PORT HERE
$WQPHPhostinfo->host='http://yoursystem.com';
$WQPHPhostinfo->port='89';
$WQuserinfo = new db2webqryuser();
// PUT YOUR DB2 Web Query authorized user and password here. In real application would probably
// get this from browser input form. For this sample though it is easier to have it here.
$WQuserinfo->user='user';
$WQuserinfo->pw='password';
?>

**HTML files**

   db2webqueryframes.html
```

PHP and DB2 Web Query Web Ser
db2webquerydrillstart2.html

<form method="GET" action="db2webquery5a.php">
<center>
<img src="/db2webqry.jpg"/>
<h1>PHP integration of Web Query SDK web services</h1>
<h2><br><br>Enter Customer Balance Due Amount:</h2>
<table>
<tr><td align="center"><input type="text" name=baldue></td></tr>
<tr><td align="center"><input type="submit"></td></tr>
</table></center>
</form>

Running the first bar chart sample

To run the first sample, you will use a URL in this form: http://<your Zend/PHP host>:<port>/db2webqueryframes.html. Enter a numeric value in the text box for balance due and press submit query. The value inputted will be used in the query selection criteria to return those accounts with balances due greater than or equal to the inputted value.
PHP integration of Web Query SDK web services

Enter Customer Balance Due Amount:

Submit Query

Figure 15 Initial web page for sample 1

Clicking on the bars in the image map will invoke the WebQueryDrill and echo the results to the frame at the bottom of the page as seen in Figure 16 Results web page.
Figure 16 Results web page
Sample 2 – Pie Chart

This sample will use DB2 Web Query pie chart reports with drill down capabilities. The needed files are:

- **WebQuery.php**
  This is the generated classes created by WSDL2PHP for invoking the DB2 Web Query web services. You can use this file if you do not want to generate it as in the Generate PHP classes for the web service section.

- **WebQueryMisc.php**
  This is an include file with some predefined variables and classes including the host name and port of the PHP server and the IBM i user id and password.

- **db2webquerydrill.php**
  This is the PHP script that invokes the WebQueryDrill web service to execute the drill down query. (see Figure 13 db2webquerydrill utility php script)

- **db2webqueryframesByState.html**
  This is the HTML page that identifies the frameset and frames for the sample application.

- **db2webqueryByStatePie.php**
  This is the main PHP script that invokes the WebQueryRunFex web services to run the report created in the Create the Balance Due By State pie chart report section. It modifies the returned HTML to change the HREFs in the image map to invoke the db2webquerydrill PHP script when the hotspots are clicked on.

The principles used in the first sample are very much the same in this sample. The only difference is the report invoked in the WebQueryRunFex section of the sample PHP. The db2webqueryByStatePie.php script is identical to the db2webquery5a.php script in the first sample. The changes are in the set up of the WebQueryRunFex object to invoke the Balance Due By State report built in Create the Balance Due by State pie chart report section. Specifically the MREdomain, MREfolder and name variables in the WebQueryRunFex object will point to the report created to generate the pie chart as seen below. To get the specific domain, folder and name refer to Figure 7 Select properties and Figure 8 Properties dialog to select the properties of the Balance Due By State report.

```php
$WQWebQueryRunFex = new WebQueryRunFex();
$WQWebQueryRunFex->IBIWS_cookie = new LogOnInfo();
$WQWebQueryRunFex->IBIWS_cookie = $_SESSION['wqloginfo'];
$WQWebQueryRunFex->IBIWS_fexinfo = new FexInfo();
$WQWebQueryRunFex->IBIWS_fexinfo->MREdomain='l0lkv6tr/l0lkv6tr.htm';
$WQWebQueryRunFex->IBIWS_fexinfo->MREfolder='#qcustcdtlmvl';
$WQWebQueryRunFex->IBIWS_fexinfo->name='app/balance_due_by_state.fex';
$WQWebQueryRunFexResult = $WQSoapClient->WebQueryRunFex($WQWebQueryRunFex);
$LinkArrayEntry = new LinkArrayEntry();
```
$LinkArrayEntry = $WQWebQueryRunFexResult->WebQueryRunFexResult->links;

**Running the second pie chart sample**

To run the second sample, you will use a URL in this form: http://<your Zend/PHP host>:<port>/db2webqueryframesByState.html. The sections of the pie chart are clickable enabling the user to drill down to run report that returns all of the customers for that pie section (state) as seen in the bottom frame.

![Image of pie chart web page](http://em15a:89/db2webqueryframesByState.html)
Appendix A. A Java framework for DB2 Web Query SDK

There is a web based solution available to help simplify the interaction and implementation with the DB2 Web Query web services. This Java servlet infrastruction handles all DB2 Web Query sdk functionality using one easy to use interface callable from many or multiple applications. This solution is meant for quick and easy SDK integration with little to no programming required, all you need to do is upload the WAR file to a Java Web environment such as the Integrated Application Server (IAS) which is integrated into the IBM i operating system.

For more information on this infrastructure please contact Tyler Even from IBM at teven@us.ibm.com.
Summary

The popularity of PHP as a premier web application development environment on the System i and the ad hoc dynamic reporting and graphic capabilities of DB2 Web Query can together provide an opportunity to create robust web applications. Using the Web Query Software Developer Kit and the examples in this paper will enable you to build your own rich web applications with web query reports.
Resources

These Web sites provide useful references to supplement the information contained in this document:

- IBM Systems Magazine: DB2 Web Query SDK Arrives (Gene Cobb)

- IBM DB2 Web Query for i
  http://ibm.com/systems/i/db2/webquery/

- WebFOCUS Web Query Web Services
  This is the IBI document describing the Web Service SDK.

- IBM DB2 Web Query Community

- IBM Systems Magazine: The Evolution of DB2 Web Query for i

- IBM Redbooks

About the author

Jon Rush is a technical consultant in ISV Business and Solution Enablement. He is a senior software engineer specializing in WebSphere, IBM Hypertext Preprocessor (PHP) and IP Telephony on the System i platform. Jon has helped hundreds of System i solution providers enhance their applications to use IBM eBusiness technologies such as IBM Net.Data®, WebSphere and PHP.
Trademark and special notices

© Copyright IBM Corporation 2008. All rights Reserved.

References in this document to IBM products or services do not imply that IBM intends to make them available in every country.

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel Inside (logos), MMX, and Pentium are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

SET and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Other company, product, or service names may be trademarks or service marks of others.

Information is provided "AS IS" without warranty of any kind.

All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics may vary by customer.

Information concerning non-IBM products was obtained from a supplier of these products, published announcement material, or other publicly available sources and does not constitute an endorsement of such products by IBM. Sources for non-IBM list prices and performance numbers are taken from publicly available information, including vendor announcements and vendor worldwide homepages. IBM has not tested these products and cannot confirm the accuracy of performance, capability, or any other claims related to non-IBM products. Questions on the capability of non-IBM products should be addressed to the supplier of those products.

All statements regarding IBM future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. Contact your local IBM office or IBM authorized reseller for the full text of the specific Statement of Direction.

Some information addresses anticipated future capabilities. Such information is not intended as a definitive statement of a commitment to specific levels of performance, function or delivery schedules with respect to any future products. Such commitments are only made in IBM product announcements.

PHP and DB2 Web Query Web Ser
information is presented here to communicate IBM's current investment and development activities as a good faith effort to help with our customers' future planning.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Photographs shown are of engineering prototypes. Changes may be incorporated in production models.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.