Data URL

Sets the URL to a server-side script that returns the data for a database-driven grid. The server-side logic can be written with Profound UI's Universal Displays or any other web development framework. The script will receive parameters "start" and "limit" in the HTTP POST data to, which tell which records in the set are to be returned. The script is called repeatedly as the user scrolls through the grid so that only one page of records is returned by the server at one time. The output is a record set in JSON format, see the examples below. On the first request to the server-side script an additional parameter "getTotal=1" is passed. When this is present, the script must return the total number of records in the set using the "totalRecs" property in the output.

The "colWidths" property that can be used in the JSON data relates to how wide the columns should be when the grid data is exported to an XLSX document. This is not intended to change the column widths for the grid widget when the data is rendered in the browser. If you need to change the column widths of the grid widget, we recommend using the grid's setProperty() API.

Promptable? YES
Possible Values: Any valid URL
Bindable? YES
Products: ProfoundUI, Genie

Example: Success

```
{
    "success": true,
    "response": {
        "results": [
            {
                "PNAME": "QUEST BOREAL",
                "PRID": "523"
            },
            {
                "PNAME": "QUEST ECLIPSE",
                "PRID": "524"
            },
            {
                "PNAME": "QUEST EQUINOX",
                "PRID": "525"
            }
        ],
        "totalRecs": 6 // Two pages, three records per page.
    }
}
```

Example: Failure

```
{
    "success": false,
    "errorId": "08001",
    "errorText": "My error message."
}
```

See External Programming and Authentication APIs for synchronizing the CGI program with the user's session.

Starting with ProfoundUI 6.7.2, when using the "load fields into widgets" property with the "load all rows" property, the limit passed is -1

PHP Example

This example assumes your installation has PHP enabled. Setup a grid in a Rich Display File or Genie screen with the following properties:

- has header: true
- column headings: PRID,PRNAME
- number of rows: 7
- number of columns: 2
- scrollbar: sliding
- data url: /mydataurlexample.php

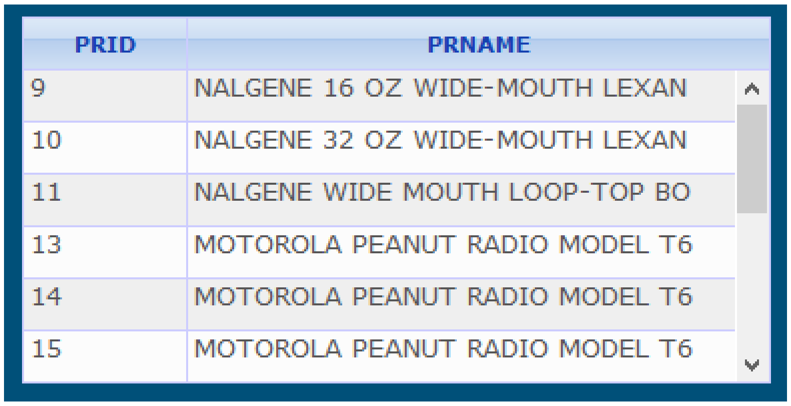
Note that the header row counts as one row, so when "number of rows" is 7, there are actually 6 rows of data.

Place the following PHP code into a file named by the "data url" in the grid; e.g. /www/profoundui/htdocs/mydataurlexample.php.

mydataurlexample.php

```
<?php
/*
Simple web service to supply a grid "data url" property.
Profound Logic Software. 2016.
*/
header('Content-Type: text/json');

// Response is bad until proven otherwise.
$response = array("success" => false);

// Setup some sample data.
$data = array();
$data[] = array("PRID" => 9, "PNAME" => "NALGENE 16 OZ WIDE-MOUTH LEXAN");
$data[] = array("PRID" => 10, "PNAME" => "NALGENE 32 OZ WIDE-MOUTH LEXAN");
$data[] = array("PRID" => 11, "PNAME" => "NALGENE WIDE MOUTH LOOP-TOP BO");
$data[] = array("PRID" => 13, "PNAME" => "MOTOROLA PEANUT RADIO MODEL T6");
$data[] = array("PRID" => 14, "PNAME" => "MOTOROLA PEANUT RADIO MODEL T6");
$data[] = array("PRID" => 15, "PNAME" => "MOTOROLA PEANUT RADIO MODEL T6");
$data[] = array("PRID" => 17, "PNAME" => "SUN SHOWER ENCLOSURE");
$data[] = array("PRID" => 18, "PNAME" => "NALGENE 16 OZ NARROW-MOUTH LEX");
$data[] = array("PRID" => 19, "PNAME" => "NALGENE 32 OZ NARROW-MOUTH LEX");
$data[] = array("PRID" => 23, "PNAME" => "POLYPRO UNDERWEAR BOTTOMS");
$data[] = array("PRID" => 24, "PNAME" => "PRINCETON TEC SPORT FLARE");
$data[] = array("PRID" => 25, "PNAME" => "PRINCETON TEC SOLO");
$data[] = array("PRID" => 26, "PNAME" => "PRINCETON TEC VORTEC");
$data[] = array("PRID" => 27, "PNAME" => "PRINCETON TEC TEC20");

// The request is valid: "start" and "limit" are at least 1.
if( isset($_POST["limit"]) && isset($_POST["start"]) && (int)$_POST["limit"] > 0 && (int)$_POST["start"] > 0 ){
    $response["success"] = true;
    $response["response"] = array("results" => array());

    // The grid sends us "getTotal" on the first request. If this is the first request, we respond with the number of
records.
    if( isset($_POST["getTotal"]) && (int)$_POST["getTotal"] == 1 ){
        $response["response"]["totalRecs"] = 14;
    }

    // Decide which section of data to read, and append data to response.
    $idx = $_POST["start"] - 1;    //PHP array index begins at 0, but POST["start"] starts at 1.
    $maxrow = min( $idx + $_POST["limit"], count($data));
    while( $idx < $maxrow ){
        $response["response"]["results"][] = $data[$idx];
        $idx++;
    }
}//end if request valid.
else
{
    $response["errorId"] = "00001";
    $response["errorText"] = "Invalid start or limit parameters.";
}
echo json_encode($response, JSON_PRETTY_PRINT );
?>
```

That's all:



PRID	PRNAME
9	NALGENE 16 OZ WIDE-MOUTH LEXAN
10	NALGENE 32 OZ WIDE-MOUTH LEXAN
11	NALGENE WIDE MOUTH LOOP-TOP BO
13	MOTOROLA PEANUT RADIO MODEL T6
14	MOTOROLA PEANUT RADIO MODEL T6
15	MOTOROLA PEANUT RADIO MODEL T6

Note, each time you scroll to a new starting row, the client browser requests a new page from the server-side script. If the parameters do not change, then the client caches responses.

HTTP POST Parameters for Other Grid Features

Find, filter, and order-by are available with data URL grids starting with Profound UI Version 5, Fix Pack 6.0.

Find and Filter with Data URL Grids

If the grid's "find option" or "filter option" properties are defined, then the grid allows a user to find or filter the grid data. The grid sends special parameters to your web service to indicate what type of find or filter. Therefore, for your grid web service to support Find/Filter, it will need to behave according to the specifications listed below.

Find

The request parameters your web service should accept are:

- findcol - The column number to search. The first column is column 1, the 2nd is 2, etc.
- findval - The text to search. Custom URL, DB-Driven, and Handler-driven grids do case-insensitive searches.
- start - Which record to begin searching on. A value of 1 means the web service should return the next match on or after the first record.
- limit - This is the number of visible rows in your grid and should always be passed in requests.

If the web service attempted to find something but nothing matched, then it should respond with this output:

```
{
    "success": true,
    "response": {
        "results": [],
        "matchRow": -1
    }
}
```

If the web service found a matching record, then it should respond with that record as the first entry in the "results" array, followed by as many subsequent, consecutive records that will fill the visible grid rows. Also, "matchRow" should be the record number where the match was found.

For example, if a grid has 4 visible rows, and the text "motorola" is first found on the 4th record, then the response might look like this:

```
{
  "success": true,
  "response": {
    "results": [
      { "PRID": "13", "PNAME": "MOTOROLA PEANUT RADIO MODEL T6"},
      { "PRID": "14", "PNAME": "MOTOROLA PEANUT RADIO MODEL T6"},
      { "PRID": "15", "PNAME": "MOTOROLA PEANUT RADIO MODEL T6"},
      { "PRID": "17", "PNAME": "SUN SHOWER ENCLOSURE"}
    ],
    "matchRow": 4
  }
}
```

If the found text is the last record, then the web service should return only that record inside the "results" JSON array.

Filter - Request

The grid supports filtering up to 10 columns. Each filtered column sends three parameters in the AJAX request. The prefixes for each column's filter are:

- fltrcol - The column number to filter. The first column would have fltrcol==1, the 2nd would have fltrcol==2, etc.
- fltrval - The text or number to filter in the column. Some filter-types use multiple values of this.
- fltrtype - What type of search to perform. The user may input filter expressions as explained in setFilter. The corresponding filter types passed to your web service are as follows:
    - "CON" - contains. Fetch data containing the filter text.
    - "BET" - between. Fetch data whose value is between two values. This type requires two **fltrval** parameters.
    - "STW" - starts with. Fetch data starting with the filter text.
    - "VAL" - values. Fetch data matching any of a comma-separated list of values. This type requires a **fltrval** parameter for each value, and **fltr cnt** must be passed.
    - "=" - Fetch data whose value is the filter text, not a partial match.
    - ">=" - Fetch data greater or equal to the filter.
    - "<=" - Fetch data less or equal to the filter.
    - "!=" - Fetch data not equal to the filter.
    - ">" - Fetch data greater than the filter.
    - "<" - Fetch data less than the filter.
- fltrcnt - Filter count. Used only for **VAL** type filter. Indicates how many filter values are included in the POST data.
- start - This parameter will be 1 when the filter is first applied. When the user scrolls down, this will be the record number of the filtered set to return.

- getTotal - Tell the web service to include "totalRecs" in the response with the total number of filtered records. The getTotal parameter will be 1 when a filter is first applied. It is not necessary to include getTotal when the user is scrolling over the filtered results.
- limit - The number of visible rows in the grid.

The suffix for each filter parameter is a number 0 to 9. The first filter has a suffix 0, the 2nd has suffix 1, etc. Filter types that require extra **fltrval** parameters will send them with the additional suffix of "_0" for the first value, "_1" for the 2nd value, etc.

Filter Request Examples

To filter all records where the second column contains the text, "abc", the HTTP POST data would include:
&fltrcol0=2&fltrtype0=CON&fltrval0=abc&start=1&getTotal=1

To filter records where the fourth column has values between 3.00 and 3.84, the POST data would include: &fltrcol0=4&fltrtype0=BET&fltrval0_0=3.
00&fltrval0_1=3.84&start=1&getTotal=1

To filter records where the first column has one of three values, "AB471", "AC904", "AV889", the POST data would include:
&fltrcol0=1&fltrtype0=VAL&fltrcnt0=3&fltrval0_0=AB471&fltrval0_1=AC904&fltrval0_2=AV889&start=1&getTotal=1

Filter - Response

Your web service should respond as it would to a normal request: return as many records specified by the "limit" parameter or as many records are in the filtered set–whichever is fewer. If the "getTotal" parameter was passed, then the response should include a "totalRecs" value.

Order By

If the grid's Sortable Columns property is true, then a user may click a column header to sort by that column. The column number should be included in the HTTP POST data value named, "order". For example, to sort by the first column, the POST data would include: &order=1

Security

The external CGI program that handles "data url" runs in a separate job from the Rich Display RPG program. Profound UI widgets pass an "AUTH" parameter that contains a session ID to external CGI programs. The "AUTH" parameter can be used to synchronize the CGI program with the user's session. (See External Programming and Authentication APIs). Universal Display programs handle SyncJob automatically. Your external CGI program should verify that the user has permission to access what data is being requested, and the SyncJob API is a way to accomplish that.

Be extremely careful when generating a SQL query string from HTTP POST data. To avoid SQL injection vulnerability, use parameter markers in your CGI program's queries, and bind the user's input to the values. Some features, including Order By and Filter, identify a column name to sort or filter. Verify that the field name is in the query's SELECT clause before using it.