Data Format Reference for FX.php

This document is intended to demonstrate the way in which FX.php returns data. FileMaker's Web Companion returns a wealth of data and FX.php is designed to return all of it. However, the wealth of data present can sometimes make it difficult to locate exactly what you would like to display. To better understand this document, please consult *DatasetDiagram.pdf* and *displayFXDataset.php* which are part of the latest FX.php release. (There are also three other displayFX files included in the release which demonstrate special cases: a database list, a list of the layouts within a database, and information about a specific layout. You may need to modify the IP address and port number specified in the files to match the individual configuration of your FileMaker Unlimited machine.) If you do not have these files, they may be obtained from http://www.iviking.org/FX.php/ or you may email the author of this document, Chris Hansen, at FX@iviking.org.

Important: The first bold section after each heading contains sample uses/syntax of the associated returned data.

Conventions used:
+ $ReturnedData is used to signify the array containing the data returned by a query. For examples of the syntax used to populate $ReturnedData, see the 'Last Functions' portion of the *FX Functions* document which accompanies *FX.php*.
+ Lines where parameter descriptions begin are indicated by a leading '>> '.

$ReturnedData['linkNext']

```
if (strlen($ReturnedData['linkNext']) < 1) {
  echo "<span class=\"medgraytext\">Next</span>\n";
} else {
  echo '<a href="'.$ReturnedData['linkNext']."\" class=\"bbody\">Next</a>\n";
}
```

The 'linkNext' section of the returned data contains a preformed link to the next set of FileMaker records based on the current query, skip count, and set size. If all records or the last group of records in the current found set are currently displayed, 'linkNext' will be empty. The code above displays the Next link if it is present, and a grayed out "Next" otherwise.

For advanced applications, the special '-foundSetParams_begin' and '-foundSetParams_end' parameters can be used to limit which parts of the current query are included in the links stored in 'linkNext' and 'linkPrevious'. Usage syntax and examples of these parameters can be found in *FXExamples.php* and the latest *FX Functions* document.

$ReturnedData['linkPrevious']

```
if (strlen($ReturnedData['linkPrevious']) < 1) {
  echo "<span class=\"medgraytext\">Prev</span>\n";
} else {
  echo '<a href="'.$ReturnedData['linkPrevious']."\" class=\"bbody\">Prev</a>\n";
}
```

The 'linkPrevious' section of the returned data contains a preformed link to the previous set of FileMaker records based on the current query, skip count, and set size. If all records or the first group of records in the current found set are currently displayed, 'linkPrevious' will be empty. The code above displays the Previous link if it is present, and a grayed out "Previous" otherwise.

For advanced applications, the special '-foundSetParams_begin' and '-foundSetParams_end' parameters can be used to limit which parts of the current query are included in the links stored in 'linkNext' and 'linkPrevious'. Usage syntax and examples of these parameters can be found in *FXExamples.php* and the latest *FX Functions* document.

$ReturnedData['foundCount']

```
echo $ReturnedData['foundCount'] . " records matched your search criteria.";
```

'foundCount' contains the *total* number of FileMaker records found that match the last query's specified criteria.

$ReturnedData['fields']

```
$counter = 0;
foreach ($ReturnedData['fields'] as $key => $value) {
  if ($counter % 2 == 0) {
    $rowColor = '#99CC99';
  } else {
    $rowColor = '#99CCCC';
  }
  echo "<tr bgcolor=\"$rowColor\">\n";
  echo '  <td align="left" class="body">' . $value['name'] . "</td>\n";
  echo '  <td align="center" class="body">' . $value['type'] . "</td>\n";
  echo '  <td align="center" class="body">' . $value['emptyok'] . "</td>\n";
  echo '  <td align="center" class="body">' . $value['maxrepeat'] . "</td>\n";
  echo "</tr>\n";
  ++$counter;
}
```

Information about the attributes of fields present on the current layout are stored in the 'fields' section of the returned data. (Additional information is present in the 'data' section of '-view' requests.) FileMaker Pro provides information about four attributes of each field: its name, its type (e.g. container, text, number, etc.), whether it can be empty, and the maximum number of repetitions.

The 'name' attribute is simply the name that was given to the field when it was created in FileMaker. If the field is present on the current layout as a related field, it will have the layout name appended to it like this: my_relationship_name::field_name. Also, you may find it less problematic when creating databases for the web, to avoid using spaces in field names; the underscore character -- e.g. my_field_name -- and "camel caps" -- e.g. myFieldName -- are common alternatives to using spaces.

Knowing a field's 'type' can aid in properly handling the returned data. For example, in *FXExamples.php*, type checks are used so that container fields (which output URLs, as opposed to the actual image) can be properly handled.

'emptyok' simply identifies whether a field may be left empty, and could be used to dynamically insert a JavaScript check for fields which cannot be left empty.

The final attribute, 'maxrepeat' will be "1" for all but repeating fields. Although a portal may contain multiple instances of a field, unless that field is a repeating field, 'maxrepeat' will be "1". (For that matter, I'm not quite sure whether *FX.php* or even the Web Companion could handle a repeating field within a portal properly.) In the case of repeating fields, the 'maxrepeat' attribute will contain the number of repetitions defined for a specific field, and **not** the number of repetitions displayed on the current layout.

Unfortunately, at the time of this writing, FileMaker Pro does not return data indicating whether a specific field is actually modifiable. This should not be an issue when dealing with a specific, known database, but *FXExamples.php* has to attempt a write to each individual field to check for this.

$ReturnedData['data']

The element of the returned data array which does most of the work is 'data'. This section has three levels of sub-arrays within it, and contains all of the data stored in the currently returned records. For additional help in visualizing how data is stored, please consult the documents referenced in the first paragraph above; *displayFXDataset.php* should be especially helpful.

```
foreach ($ReturnedData['data'] as $key => $value) {
  if ($counter % 2 == 0) {
    $tempBGColor = '#FFFFFF';
  } else {
    $tempBGColor = '#CCCCCC';
  }
  $counter2 = 0;
  $IDsArray = explode('.', $key);
  .
  .
  .
```

The top level sub-array contains one element for each record returned. In most cases, the index for this section is FileMaker's internal record ID, a period, and then FileMaker's internal modification ID. You'll need to extract the record ID of an individual record in order to edit that record. In the example above, I use php's explode() function to split the index into its component parts. I could then access the record ID, for example, by using $IDsArray[0]. Special cases like database lists, layout lists, or layout information use a zero, a period, and a simple ordinal number as the index.

```
  print_r($value['Object_Components']);
```

Field names are used as indexes for the middle sub-array within 'data'. The preceding example -- which is commented out in *FXExamples.php* -- will display all of the contents of the Object_Components field, which has four repetitions, from the current record. (Using print_r() rather than echo() is important since the data output by *FX.php* for an entire field -- even a non-repeating field -- is always an array. Using echo() would simply display the word "Array" in this case.) Of course, displaying the contents of a field in this way is of limited utility. The lowest level sub-array is usually the best way to display a field's contents.

```
  echo($value['Object_Components'][0]);
```

At the lowest level, the data for each iteration of a field uses numbers (starting from zero) for indexes. This means that, except when dealing with repeating fields or portals, fields are accessed with a trailing "[0]" as above. (As in the previous example, this line of code is commented out in *FXExamples.php*.) In the case of repeating fields and portals, I find that php's foreach() construct is the easiest way to step through the values; please see *FXExamples.php* for a detailed example of how this is done.

$ReturnedData['URL']

```
echo $ReturnedData['URL'];
```

In most cases, the URL stored in this section can be used as an alternate way of accessing the same FileMaker Pro XML currently being parsed by *FX.php*. Because GET queries have a limited maximum length (which varies based on server software and the browser in use), complex queries may be truncated. Barring such cases, this URL can be very useful for debugging. (Tip: accessing XML in this way is the only task for which I prefer to use IE.)

$ReturnedData['errorCode']

```
echo $ReturnedData['errorCode'];
```

The 'errorCode' element of the returned data array contains any error codes that may have been returned by FileMaker Pro. A zero indicates no error; for other errors, consult FileMaker Pro's documentation, or have a look at *FMErrors.php* -- a handy file included in the latest release of *FX.php* (thanks to Gjermund Thorsen for doing the legwork.) *FXExamples.php* demonstrates one way in which this file can be integrated into a solution.

$ReturnedData['valueLists']

```
print_r($ReturnedData['valueLists']);
print_r($ReturnedData['valueLists']['myValueList']);
echo($ReturnedData['valueLists']['myValueList'][0]);
```

When an FMView() action is called, any value lists on the layout specified will be returned in this element of the returned data array. Please note that this element is **only** returned from an FMView() action; you may wish to perform a separate FMView() to pull the value lists for the layout in your current search request. The 'valueLists' section contains two sub-arrays. The first sub-array uses the value lists' names as indexes, while the second is numbered from zero. A typical use might be to step through the desired value list using php's foreach() contruct. Please see *FXExamples.php* for a complex demonstration of how the contents of this element can be used.

Additional Reading

For more information about FX.php (including the latest version of this document):

http://www.iviking.org/

For more information on FileMaker and XML:

http://www.filemaker.com/downloads/pdf/xml_Chapter_7.pdf

For more information on the parameters accepted by FileMaker's Web Companion:

http://www.filemaker.com/downloads/pdf/xml_Appendix_B.pdf

For additional information on the error codes returned by FileMaker's Web Companion:

http://www.filemaker.com/downloads/pdf/xml_Appendix_C.pdf

For additional information about PHP, see the online manual at:

http://www.php.net/manual/en/

or look for books on PHP or FileMaker at your local bookseller.