SurveyResponse Return Per Question Type

Last Modified on 09/17/2020 11:20 am EDT |

This document includes specifics of the SurveyResponse return for all fields in Alchemer that collect data, including all question types, other textboxes, other row headers and comment fields. IDs and reporting values are essential for identifying survey response data; the Survey Legend will be your friend in this process.

Jump to a Question Type

Text Fields and Single-Select Questions

Text fields and single-select questions have the same return, which includes the question ID and the response. The response will be the reporting value, in the case of single-select questions, or the text that was entered by the respondent in the case of text fields.

Text fields and single-select questions include: Radio Buttons, Dropdown Menu, Textbox, Essay, Email, Date, Slider, Net Promoter Score®, Rating, Number, Percent, Image Select (select one). In addition, all of these question types, when part of a Custom Group or Contact Form, return in the same format; learn more.

```
[[question(ID)]] => Response
```

debug example:

```
[[question(2)]] => 1
```

xml example:

```
1
```

json example:

```
"[question(2)]":"1",
```

pson example:

```
s:13:"[question(2)]";s:1:"1";
```

Radio Button Grid

The Radio Button Grid is a series of Radio Button questions where each row is a question. Each row has its own question ID and the response will be the reporting value.

```
[[question(ID)]] => Response
[[question(ID)]] => Response
```

debug example:

```
[[question(3)]] => 4
[[question(4)]] => 5
```

xml example:

```
4
5
```

json example:

```
"[question(3)]":"4",
"[question(4)]":"5",
```

pson example:

```
s:13:"[question(3)]";s:8:"4";
s:13:"[question(4)]";s:8:"5";
```

Checkboxes and Image Select (multi select)

Multi-select questions, which includes Checkboxes and Image Select (multi select), will return multiple fields. Think of each answer option as a separate question where the answer can be selected or unselected. The return includes the question ID, the option SKU and the response. The response will be the reporting value.

```
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] =>
```

If the option was not selected the response will be empty.

debug example:

```
[[question(2), option(10001)]] => 1
[[question(2), option(10002)]] =>
```

xml example:

```
1
```

json example:

```
"[question(2), option(10001)]":"1",
"[question(2), option(10002)]":"",
```

pson example:

```
s:28:"[question(2), option(10001)]";s:1:"1";
s:28:"[question(2), option(10002)]";s:0:"";
```

Checkbox Grid

The Checkbox Grid is a series of Checkbox questions where each row is a question. Because checkbox questions are multi-select questions each answer option is treated as a separate question where the answer can be selected or unselected. For this reason, the Checkbox Grid returns rows x columns elements that can be identified by the question ID (row) and the option SKU (column) combination. The response will be the reporting value.

For example, a Checkbox Grid with 3 rows and 3 columns will out put 9 fields.

```
[[question(ID), option(SKU)]] =>
[[question(ID), option(SKU)]] =>
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] =>
[[question(ID), option(SKU)]] =>
[[question(ID), option(SKU)]] =>
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] =>
```

If the option was not selected the response will be empty.

debug example:

```
[[question(3), option(10001)]] =>
[[question(3), option(10002)]] =>
[[question(3), option(10003)]] => 3
[[question(4), option(10001)]] => 1
[[question(4), option(10002)]] =>
[[question(4), option(10003)]] =>
[[question(5), option(10001)]] =>
[[question(5), option(10002)]] => 2
[[question(5), option(10003)]] =>
```

xml example:

```
<response_data>
3
1
</response_data>
<response_data>
2
</response_data>
```

json example:

```
"[question(3), option(10001)]":"",
"[question(3), option(10002)]":"",
"[question(3), option(10003)]":"3",
"[question(4), option(10001)]":"1",
"[question(4), option(10002)]":"",
"[question(4), option(10003)]":"",
"[question(5), option(10001)]":"",
"[question(5), option(10002)]":"2",
"[question(5), option(10003)]":"",
```

pson example:

```
s:28:"[question(3), option(10001)]";s:0:"";
s:28:"[question(3), option(10002)]";s:0:"";
s:28:"[question(3), option(10003)]";s:1:"3";
s:28:"[question(4), option(10001)]";s:1:"1";
s:28:"[question(4), option(10002)]";s:0:"";
s:28:"[question(4), option(10003)]";s:0:"";
s:28:"[question(5), option(10001)]";s:0:"";
s:28:"[question(5), option(10002)]";s:1:"2";
s:28:"[question(5), option(10003)]";s:0:"";
```

Star Rating Grid

The Star Rating Grid returns rows x columns elements that can be identified by the question ID (row) and the option SKU (column) combination. The response will be the count of stars selected.

```
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] => Response
```

debug example:

```
[[question(3), option(10001)]] => 5
[[question(3), option(10002)]] => 2
[[question(3), option(10001)]] => 4
[[question(3), option(10002)]] => 1
```

xml example:

```
5
2
4
1
```

json example:

```
"[question(3), option(10001)]":"5",
"[question(3), option(10002)]":"2",
"[question(4), option(10001)]":"4",
"[question(4), option(10002)]":"1",
```

pson example:

```
s:28:"[question(3), option(10001)]";s:1:"5";
s:28:"[question(3), option(10002)]";s:1:"2";
s:28:"[question(4), option(10001)]";s:1:"4";
s:28:"[question(4), option(10002)]";s:1:"1";
```

File Upload

The File Upload question returns an array per file with the following fields:

- filedata
- filename
- filelocation
- filesize
- filetype

```
[[question(ID), option(SKU)]] => vDBFileValue Object
        (
            [filedata] =>
            [filename] => 246-0364ca7fdae2fe24df8f17ace3699968_Image.png
            [filelocation] => S3
            [filesize] => 15
            [filetype] => image/png
        )
```

Debug example:

```
[[question(2), option(10004)]] => vDBFileValue Object
        (
            [filedata] =>
            [filename] => 243-0364ca7fdae2fe24df8f17ace3699968_Summer.png
            [filelocation] => S3
            [filesize] => 95
            [filetype] => image/png
        )
```

xml example:

```
<filedata></filedata>
<filename>243-0364ca7fdae2fe24df8f17ace3699968_Summer.png</filename>
<filelocation>S3</filelocation>
<filesize>95</filesize>
<filetype>image\/png</filetype>
```

json example:

```
"[question(2), option(10004)]":{
  "filedata":"",
  "filename":"243-0364ca7fdae2fe24df8f17ace3699968_Summer.png",
  "filelocation":"S3",
  "filesize":"95",
  "filetype":"image\/png"
}
```

You can construct the link to the file with this information. All file upload links are a specific file page with this base file path:

```
http://surveygizmoresponseuploads.s3.amazonaws.com/fileuploads/
```

In addition to the base, you'll need to add the following:

- Your account ID
- Survey ID
- Name of the file

The complete link contains these elements:

```
/[account id]/[survey id ]/[filename]
```

Drag & Drop Ranking and Ranking Grid

Ranking questions, both the Drag & Drop Ranking and the Ranking Grid, will return an element per row with the question ID, the option SKU, and the response. The response is the numeric rating selected for that option.

```
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] => Response
```

debug example:

```
[[question(2), option(10001)]] => 2
[[question(2), option(10002)]] => 3
[[question(2), option(10003)]] => 1
```

xml example:

```
2
3
1
```

json example:

```
"[question(2), option(10001)]":"2",
"[question(2), option(10002)]":"3",
"[question(2), option(10003)]":"1",
```

pson example:

```
s:28:"[question(2), option(10001)]";s:1:"2";
s:28:"[question(2), option(10002)]";s:1:"3";
s:28:"[question(2), option(10003)]";s:1:"1";
```

Signature

The typed text from the Signature question is available via the API. The response will be the text that was entered by the respondent.

Dropdown Menu List

The Dropdown Menu List is a series of Dropdown Menu questions where each row is a question. Each row has its own question ID and the response will be the reporting value of the option selected by the respondent.

```
[[question(ID)]] => Response
[[question(ID)]] => Response
```

debug example:

```
[[question(3)]] => 1
[[question(4)]] => 2
```

xml example:

```
1
2
```

json example:

```
"[question(3)]":"1",
"[question(4)]":"2",
```

pson example:

```
s:13:"[question(3)]";s:1:"1";
s:13:"[question(4)]";s:1:"2";
```

Textbox List

Textbox List questions will return an element per row with the question ID, the option SKU, and the response. The response will be the text that was entered by the respondent.

```
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] => Response
```

debug example:

```
[[question(2), option(10001)]] => textbox list response
[[question(2), option(10002)]] => textbox list response
```

xml example:

```
textbox list response
textbox list response
```

json example:

```
"[question(2), option(10001)]":"textbox list response",
"[question(2), option(10002)]":"textbox list response",
```

pson example:

```
s:28:"[question(2), option(10001)]";s:21:"textbox list response";
s:28:"[question(2), option(10002)]";s:21:"textbox list response";
```

Continuous Sum

The Continuous Sum will return an element per row with the question ID, the option SKU, and the response. The response will be the value that was entered by the respondent.

```
[[question(ID), option(10001)]] => Response
[[question(ID), option(10002)]] => Response
```

debug example:

```
[[question(2), option(10001)]] => 100
[[question(2), option(10002)]] => 200
```

xml example:

```
100
200
```

json example:

```
"[question(2), option(10001)]":"100",
"[question(2), option(10002)]":"200",
```

pson example:

```
s:28:"[question(2), option(10001)]";s:3:"100";
s:28:"[question(2), option(10002)]";s:3:"200";
```

Slider List

The Slider List is a series of Slider questions where each row is a question. The return will include the question ID, the option SKU for the row, and the response, which is the value selected by the respondent.

```
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] => Response
```

debug example:

```
[[question(2), option(10001)]] => 58
[[question(2), option(10002)]] => 44
```

xml example:

```
58
44
```

json example:

```
"[question(2), option(10001)]":"58",
"[question(2), option(10002)]":"44",
```

pson example:

```
s:28:"[question(2), option(10001)]";s:2:"58";
s:28:"[question(2), option(10002)]";s:2:"44";
```

Cascading Dropdown Menu

The Cascading Dropdown Menu will return an element per menu with the question ID, the option SKU, and the response. The response will be the option that was selected by the respondent.

```
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] => Response
```

debug example:

```
[[question(2), option(10001)]] => VW
[[question(2), option(10002)]] => Golf
[[question(2), option(10003)]] => 2011
```

xml example:

```
VW
Golf
2011
```

json example:

```
"[question(2), option(10001)]":"VW",
"[question(2), option(10002)]":"Golf",
"[question(2), option(10003)]":"2011",
```

pson example:

```
s:28:"[question(2), option(10001)]";s:2:"VW";
s:28:"[question(2), option(10002)]";s:4:"Golf";
s:28:"[question(2), option(10003)]";s:4:"2011";
```

Dropdown Menu Grid

The Dropdown Menu Grid returns rows x columns elements that can be identified by the question ID (row) and the option SKU (column) combination. The response will be the option the respondent selected.

```
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] => Response
[[question(ID), option(SKU)]] => Response
```

debug example:

```
[[question(3), option(10001)]] => choice 1
[[question(3), option(10002)]] => choice 2
[[question(4), option(10001)]] => choice 3
[[question(4), option(10002)]] => choice 2
```

xml example:

```
choice 1
choice 2
choice 3
choice 2
```

json example:

```
"[question(3), option(10001)]":"choice 1",
"[question(3), option(10002)]":"choice 2",
"[question(4), option(10001)]":"choice 3",
"[question(4), option(10002)]":"choice 2",
```

pson example:

```
s:28:"[question(3), option(10001)]";s:8:"choice 1";
s:28:"[question(3), option(10002)]";s:8:"choice 2";
s:28:"[question(4), option(10001)]";s:8:"choice 3";
s:28:"[question(4), option(10002)]";s:8:"choice 2";
```

Textbox Grid

The Textbox Grid returns rows x columns elements that can be identified by the question ID (row) and the option SKU (column) combination. The response will be the text the respondent entered.

```
[[question(ID), option(10090)]] => textbox grid response
[[question(ID), option(10091)]] => textbox grid response
[[question(ID), option(10090)]] => textbox grid response
[[question(ID), option(10091)]] => textbox grid response
```

debug example:

```
[[question(3), option(10001)]] => textbox grid response
[[question(3), option(10002)]] => textbox grid response
[[question(4), option(10001)]] => textbox grid response
[[question(4), option(10002)]] => textbox grid response
```

xml example:

```
textbox grid response
textbox grid response
textbox grid response
textbox grid response
```

json example:

```
"[question(3), option(10001)]":"textbox grid response",
"[question(3), option(10002)]":"textbox grid response",
"[question(4), option(10001)]":"textbox grid response",
"[question(4), option(10002)]":"textbox grid response",
```

pson example:

```
s:28:"[question(3), option(10001)]";s:21:"textbox grid response";
s:28:"[question(3), option(10002)]";s:21:"textbox grid response";
s:28:"[question(4), option(10001)]";s:21:"textbox grid response";
s:28:"[question(4), option(10002)]";s:21:"textbox grid response";
```

Max Diff

In the SurveyResponse return for the Max Diff question type, the values assigned to each attribute based on how that individual respondent ranked the values are exported. They range from 0 to 100. These values are *not* used in computing the aggregate bayesian average and rank order you see in the Summary Reports. Think of these as a snapshot of the rank order for that response alone. To learn more about this check out our Max Diff Tutorial.

```
[[question(ID), option(SKU)]] => Value
[[question(ID), option(SKU)]] => Value
[[question(ID), option(SKU)]] => Value
[[question(ID), option(SKU)]] => Value
```

debug example:

```
[[question(2), option(10001)]] => 14.2857
[[question(2), option(10002)]] => 33.3333
[[question(2), option(10003)]] => 50
[[question(2), option(10004)]] => 80
```

xml example:

```
14.2857
33.3333
50
80
```

json example:

```
"[question(2), option(10001)]":"14.2857",
"[question(2), option(10002)]":"33.3333",
"[question(2), option(10003)]":"50",
"[question(2), option(10004)]":"80"
```

pson example:

```
s:28:"[question(2), option(10001)]";s:7:"14.2857";
s:28:"[question(2), option(10002)]";s:7:"33.3333";
s:28:"[question(2), option(10003)]";s:2:"50";
s:28:"[question(2), option(10004)]";s:2:"80"
```

There is more detailed raw data that is reported in a [[variable(ID)]] field. This data is formatted as follows:

Shown Sku; Shown Sku; Shown Sku; Shown Sku: Best Selected Sku /Worst Selected Sku

Semantic Differential

```
[[question(ID)]] => Response
[[question(ID)]] => Response
```

debug example:

```
[[question(3)]] => 2
[[question(4)]] => 1
```

xml example:

```
2
1
```

json example:

```
"[question(3)]":"2",
"[question(4)]":"1",
```

pson example:

```
s:13:"[question(3)]";s:1:"2";
s:13:"[question(4)]";s:1:"1";
```

Contact Form and Custom Group

All Contact Form and Custom Group subquestions, the fields that actually collect data, have their own IDs and return as they would if they were not part of a grouping.

Custom Table

Custom Table data is unique in many ways. First, unlike other grid questions, where the rows are the subquestions, in custom tables, the columns are the subquestions. Each column of a Custom Table has its own question ID. Custom Tables also make use of our piping functionality to create the fields in the cells of each row.

Custom Table Radio Buttons and Dropdown Menus

Custom Table Radio Button and Dropdown Menus will return a field per row. Each field returned will include the subquestion ID (column), the row it was piped for, and the response. The response is the Radio Button or Dropdown Menu option selected.

```
[[question(ID), question_pipe("Row Header")]] => Response
[[question(ID), question_pipe("Row Header")]] => Response
```

debug example:

```
[[question(3), question_pipe("Row 1")]] => 2
[[question(3), question_pipe("Row 2")]] => 1
```

xml example:

```
2
1
```

json example:

```
"[question(3), question_pipe(\"Row 1\")]":"2",
"[question(3), question_pipe(\"Row 2\")]":"1",
```

pson example:

```
s:37:"[question(3), question_pipe("Row 1")]";s:1:"2";
s:37:"[question(3), question_pipe("Row 2")]";s:1:"1";
```

Custom Table Checkboxes

Custom Table Checkbox questions return number of rows x number of option elements. Think of each answer option as a separate question where the answer can be selected or unselected. The return includes the question ID for the subquestion, the option SKU, the row header from the Custom Table, and the response. The response will be option selected.

```
[[question(ID), option(SKU), question_pipe("Row Header")]] => Response
[[question(ID), option(SKU), question_pipe("Row Header")]] =>
```

debug example:

```
[[question(3), option(10001), question_pipe("Row 1")]] => 1
[[question(3), option(10002), question_pipe("Row 1")]] => 2
[[question(3), option(10001), question_pipe("Row 2")]] =>
[[question(3), option(10002), question_pipe("Row 2")]] =>
```

xml example:

```
1
2
```

json example:

```
"[question(3), option(10001), question_pipe(\"Row 1\")]":"1",
"[question(3), option(10002), question_pipe(\"Row 1\")]":"2",
"[question(3), option(10001), question_pipe(\"Row 2\")]":"",
"[question(3), option(10002), question_pipe(\"Row 2\")]":"",
```

pson exmple:

```
s:52:"[question(3), option(10001), question_pipe("Row 1")]";s:1:"1";
s:52:"[question(3), option(10002), question_pipe("Row 1")]";s:1:"2";
s:52:"[question(3), option(10001), question_pipe("Row 2")]";s:0:"";
s:52:"[question(3), option(10002), question_pipe("Row 2")]";s:0:"";
```

Custom Table Essay and Textbox Fields

Custom Table Essay and Textbox fields will return a field per row +1. The first returned field will be empty. The second and later elements will include the question ID for the subquestion, the row header from the Custom Table, and the response. The response will be the text that was entered

by the respondent.

```
[[question(ID)]] =>
[[question(ID), question_pipe("Row Header")]] => Response
[[question(ID), question_pipe("Row Header")]] => Response
[[question(ID)]] =>
[[question(ID), question_pipe("Row Header")]] => Response
[[question(ID), question_pipe("Row Header")]] => Response
```

debug example:

```
[[question(3)]] =>
[[question(3), question_pipe("Row 1")]] => Custom Table Textbox Response
[[question(3), question_pipe("Row 2")]] => Custom Table Textbox Response
[[question(4)]] =>
[[question(4), question_pipe("Row 1")]] => Custom Table Essay Response
[[question(4), question_pipe("Row 2")]] => Custom Table Essay Response
```

xml example:

```


Custom Table Textbox Response
Custom Table Textbox Response

Custom Table Essay Response
Custom Table Essay Response
```

json example:

```
"[question(3)]":"",
"[question(3), question_pipe(\"Row 1\")]":"Custom Table Textbox Response",
"[question(3), question_pipe(\"Row 2\")]":"Custom Table Textbox Response",
"[question(4)]":"",
"[question(4), question_pipe(\"Row 1\")]":"Custom Table Essay Response",
"[question(4), question_pipe(\"Row 2\")]":"Custom Table Essay Response",
```

pson example:

```
s:13:"[question(3)]";s:0:"";
s:37:"[question(3), question_pipe("Row 1")]";s:29:"Custom Table Textbox Response";
s:37:"[question(3), question_pipe("Row 2")]";s:29:"Custom Table Textbox Response";
s:13:"[question(4)]";s:0:"";
s:37:"[question(4), question_pipe("Row 1")]";s:27:"Custom Table Essay Response";
s:37:"[question(4), question_pipe("Row 2")]";s:27:"Custom Table Essay Response";
```

Other Textboxes

Other Textboxes can be added to both Radio Button and Checkbox questions. The return will include two elements. The first element will be for the selected option. The return will include the question ID and the response, which is a reporting value. The second element will include the question ID, the option sku, and the response, which is the text the respondent entered in the text field.

```
[[question(ID)]] => Response
[[question(ID), option("SKU-other")]] => Response
```

debug example:

```
[[question(2)]] => Other, Please specify
[[question(2), option("10003-other")]] => Something else
```

xml example:

```
Other, Please specify
Something else
```

json example:

```
"[question(2)]":"Other, Please specify",
"[question(2), option(\"10003-other\")]":"Something else",
```

pson example:

```
s:13:"[question(2)]";s:16:"Other - Write In";s:36:"[question(2), option("10003-other")]";s:14:"something else";
```

Other Row Header

Other Row Headers can be added to the following grid questions to collect other-specify responses:

- Radio Button Grid
- Checkbox Grid
- Dropdown Menu Grid
- Textbox Grid
- Star Rating Grid

Other row headers will return the question ID, the text the survey respondent entered in the other row text field, and grid response.

```
[[question(ID), question_pipe("Row Header Response")]] => Response
```

debug example:

```
[[question(2), question_pipe("Google")]] => Sometimes
```

xml example:

```
Sometimes
```

json example:

```
"[question(2), question_pipe(\"Google\")]":"Sometimes",
```

pson example:

```
s:39:"[question(2), question_pipe("Google+")]";s:9:"Sometimes";
```

Comments

Comments can be added to any question within the survey. They will appear at the bottom of the SurveyResponse return and will have an ID that corresponds to the question ID with which it is associated.

```
[[comment(ID)]] => Response
```

debug example:

```
[[comment(2)]] => Excellent service and atmosphere
```

xml example:

```
Excellent service and atmosphere.
```

json example:

```
"[comment(2)]":"Excellent service and atmosphere"
```

pson example:

```
s:12:"[comment(2)]";s:33:"Excellent service and atmosphere.";
```