



SurveyToGo Scripting Best Practices

The screenshot shows the SurveyToGo Studio interface. On the left, there's a sidebar with a tree view of projects and surveys, including 'Marky Mark' and 'Airport List Demo Survey'. The main window displays the 'Designer: Airport List Demo S...' tab. It shows a survey tree with various questions and their descriptions. In the center, there are two tabs: 'Advanced Scripts' and 'Question Start Script'. The 'Question Start Script' tab contains the following JavaScript code:

```
var a = "Test";
if (a == "Test") {
    // Do something
}
else {
    // Do somethings else
}
```

Below this is a 'Question End Script' tab with a single line of code '1'. On the right side of the interface, there's a 'Preview [PC] - Airport List Demo Survey*' section showing a smartphone screen with a survey question: 'What colors do you like?' followed by a list of color options: Red, Green, Blue, Yellow, Other1, and Other2. There are also buttons for 'Run In Emulator' and 'Preview Settings'. At the bottom of the interface, there are 'SUPPORT TICKET' buttons.



Table of Content

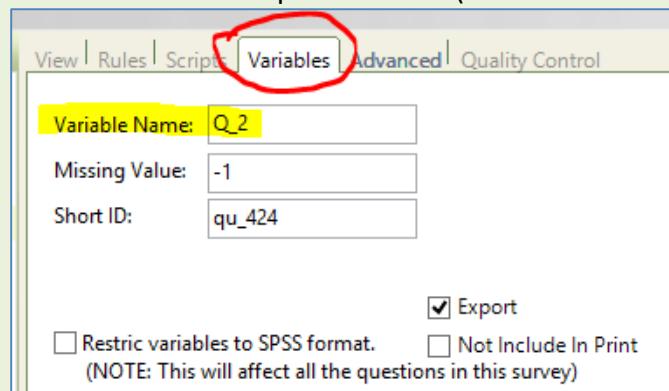
1	OVERVIEW.....	3
2	VARIABLE NAMES.....	3
3	SHORT IDS DISPLAY	4
4	ANSWER SCALES.....	5
5	GENERIC CODE WRITING	5
6	MULTIPLE FILTERING METHODS	6
7	DUMMY QUESTIONS	7
8	EXPRESSION QUESTIONS	8
9	RANDOM CALCULATIONS	9
10	REMEMBER THAT TEXT PIPING IS FOR DISPLAY ONLY	9
11	THE IMPACT OF BACK	10
12	DETERMINE IF A QUESTION WAS ANSWERED.....	10
13	ANSWER CODES.....	11
14	TRACKER PROJECTS – KEEP VARIABLE NAMES CONSISTENT	13
15	TRACKER PROJECTS – KEEP CODES CONSISTENT	13
16	TRACKER PROJECTS – UPDATE LISTS WITHOUT REMOVING / OVERRIDING ITEMS	14
17	“NULL RESPONSE” QUESTIONS.....	16
18	“NULL RESPONSE” ANSWERS/TOPICS.....	16
19	REVIEW THE DATA BEFORE GOING LIVE	16
20	EXPORTED QUESTIONS	17

1 Overview

CAPI surveys are dynamic and can contain complex logic in different levels. SurveyToGo offers extended scripting methods that provide solutions to different yet common surveying scenarios. The next article will present the scripting best practices for you to build the most robust and efficient scripts while saving time during scripting and avoiding production and/or data structure related mistakes.

2 Variable names

It is important to set a unique variable name for each question in the script. The variable name is generated automatically once creating the question yet it is important to manually change it (through the question's Variables tab) to be the same as the variable name / question number that is mentioned in the questionnaire (those variable names are later shown in the data):



If in the questionnaire there're several questions with the same variable name (which happens sometimes if by mistake or not), the client needs to be informed and be asked to change the questionnaire so that each question will have a unique variable name.

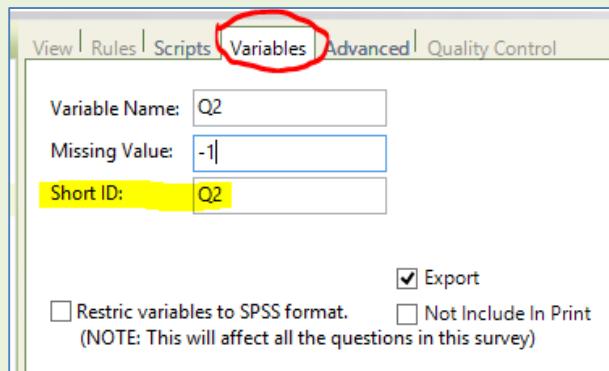
In some cases, for logical reasons, there will be additional questions in the script that don't exist in the questionnaire (such as dummy questions, or visible questions that were created to answer some structural needs). It is recommended to manually set unique variable names for those as well rather than leave those with the default names.

Make sure to set a valid variable name (due to SPSS limitations):

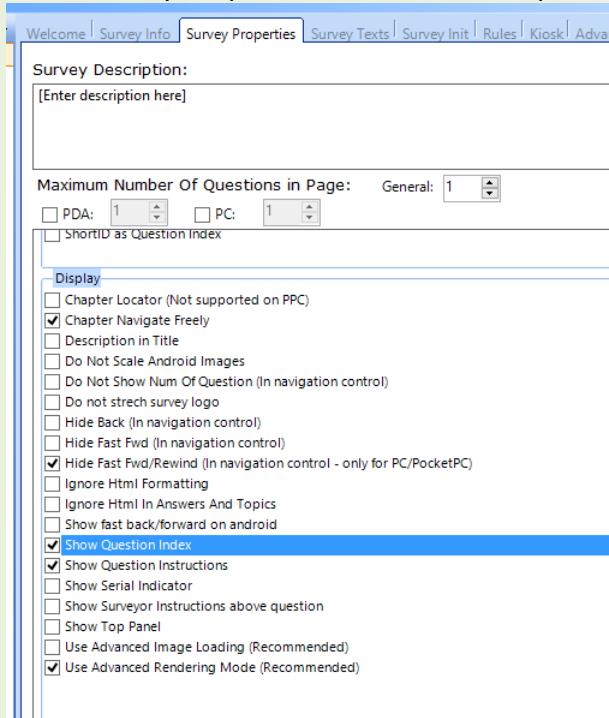
1. Always start with a letter
2. No white characters (such as spaces...)
3. No special characters of any sort
4. Only regular letters and numbers, underscores are allowed as well.

3 Short IDs display

In a survey script, each question has an ID called "Short ID", in addition to the Variable name. Unlike the Variable name, the Short ID doesn't have to be unique, but it is common to set it the same as the Variable name:

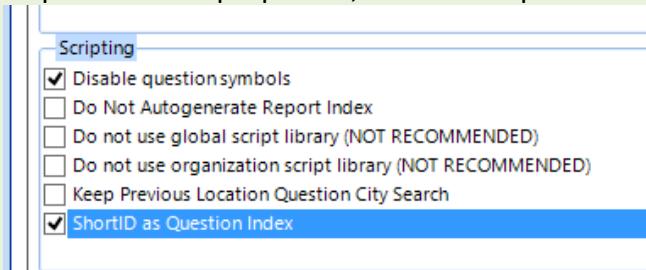


In the Survey Properties tab, there's an option to display the question's index on screen:



That means the index can be displayed as part of the question's body text, at the start of it.

As part of those properties, there's an option to define the "Short ID" as the question's index:



That means that, if chosen both to display the question's index and to use the Short ID as the question's index, the Short ID itself will be displayed on screen in the question's body text.

This is useful to determine what the displayed question on the script is, and this is needed most often.

4 Answer Scales

It is most often that a certain answers list appears in several questions on the questionnaire. In this case it is best to define that list as an Answer Scale – this scale is easily defined once when entering all the answers, and once it is created you can choose to use that scale in all relevant questions. The benefits:

- Enter the answers to the script just once
- If there's any structural / textual change requested for that list after it was created, you can apply this change once on the scale itself instead of performing it several times in all questions containing that list.
- Making the script clear and readable once you see all questions using the same scale
- Easy way to implement loop iterations when needed to iterate on answers/topics.

5 Generic Code WRITING

Always do your best to build your script in the most generic way as possible. That is, avoid writing hard-coded values in the script itself that can later change or won't fit when re-using this script in other places.

For example: it is very common that an answers list of some brands will contain a "None of the above" option as the last answer in the list, let's say this is option index #10. In a lot of cases this "None of the above" option will require a "special treatment" or some unique reference as this



Scripting Best Practices Guide

option is exceptional in this list of brands. For instance, there can be a question that the brands are filtered from a previous question but the “None” option should be displayed in any case. So you would want to use a function that will make sure that option is visible even after performing the filter, such as:

```
SetAnswerVisible(CurrQues, true, false, 10);
```

The “10” parameter is a hard-coded value. If in some stage the index of the “None of the above” option in this question will be changed to something else (for instance a new brand option will be added to the list and be placed before that last option...), the code above will have to be manually modified, so that instead of “10” it will have the new index of the “None of the above” option. To avoid mistakes and redundant work in updated the script, instead of writing “10” you can call a function that will automatically return the answers amount in the list. This function is called “GetAnswerCount(QuestionIndex)”. So, as the “None of the above” option is always placed last in the list, its index equals to the total answers amount in the list. So the right way to write this code is this:

```
SetAnswerVisible(CurrQues, true, false, GetAnswerCount(CurrQues));
```

This keeps it dynamic and generic and will always refer to the last answer in the list (the “None of the above”), even if its index will be changed.

6 Multiple Filtering Methods

The most common filtering request in questionnaires is when you have 2 questions with the same answers list, and you want to filter the answers in the second question based on the chosen answers in the first question. That is – in the second question, show only the answers that were chosen in the first question. The way to do so is to use this function:

```
FilterAnswersByAnswers(QRef(2), QRef(1));
```

There're cases where you'll need to perform several separate filtering commands in one question. For example: Q1, Q2 and Q3 all have the same answers list. This is a brands list of 9 brands, and the 10th answer is “None”. The filtering logic for Q3 is as follows:

- Display only the brands that were chosen in Q1
- From those displayed brands – hide those that were chosen in Q2
- Also display the “None” option.

You cannot perform all of the above in one function call, you'll need to do it in stages, in Q3 start script:

- First, display all answers coded in Q1 this way:

```
FilterAnswersByAnswers(CurrQues, QRef(1));
```

- Now we want to hide the answers that were chosen in Q2. In most cases we would use this

overload function this way:

```
FilterAnswersByAnswers(CurrQues, false, QRef(2));
```

But, this would “cancel/reset” the first filtering call, because this function says: “hide only the answers that were chosen in Q2 and show all the rest”. Meaning, an answer that was hidden after the first filtering call (because it was not chosen in Q1), will now be shown after the second filtering call and that’s not what we want (we want to hide the answers chosen in Q2 within the displayed answers that were chosen in Q1).

To overcome this, use the third overload for this filtering method:

```
FilterAnswersByAnswers(CurrQues, false, false, QRef(2));
```

The extra “false” parameter (the third parameter sent to this function) is the “Reset” parameter. “False” means not to reset previous answers filtering, if made. So in this case, an answer that was hidden after the first filtering call (because it was not chosen in Q1) will stay hidden after the second filtering call as we wish.

- Now, we can perform a similar trick to achieve the third rule: To show the “None” option without resetting any previous filtering, use this overload for the “SetAnswerVisible()” function:

```
SetAnswerVisible(CurrQues, true, false, 10);
```

That means: “Display answer index 10, but do not reset any previous filtering”.

7 Dummy questions

“Dummy questions” are questions that are hidden from the surveyor. That means these questions are part of the script but they’re not displayed on screen.

Dummy questions are very helpful and can simplify the implementation in some cases. It is highly recommended to use dummy questions where you can.

Common scenarios for dummy questions use:

- Questions Q1-Q4 contain the same answer scale. Q5 is a question inside a loop chapter, and the needed logic is to iterate on all the answers that were coded through Q1-Q4. So, instead of generating a long iteration entrance rule referring to Q1-Q4, it will be better to create a dummy multiple selection question (could be placed after Q4) with the same answer scale, auto-set it with all the answers coded through Q1-Q4 (the recommended way would be to write the setting code in an expression question to make sure the settings happens once in one place), and refer only to that question in the iteration entrance rule (using the “Contains” function...)
- Q1 is a multiple selection question contains an answer scale. Q2-Q5 contain the same answer scale. The needed logic is to randomly choose up to 5 answers from all answers that



Scripting Best Practices Guide

were coded in Q1, and show only those 5 answers in Q2-Q5. In this case you would want to choose those 5 answers randomly, save those (so later in the exported data you'll be able to determine what were the 5 chosen ones) and filter the answers in Q2-Q5 so that only those 5 answers will be displayed.

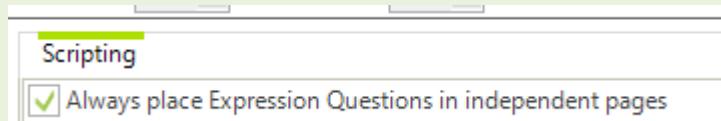
The best way would be to create a dummy question after Q1, and generate a code in Q1 end script that will set 5 random answers in that dummy question. Once the 5 answers are set in the dummy question, those will be shown in the data clearly, and you'll be able to use the simple "FilterAnswersByAnswers()" function in Q2-Q5, that will filter from that dummy.

8 Expression Questions

The use of expressions questions is very common and useful in scripting, where you need to perform some calculations, save certain values and / or perform some actions at a certain point in the survey.

As an expression question isn't displayed on screen, the way it works is that it is "joined" to another question's screen (the visible question before the expression or the visible question after the expression). Due to that behavior, the code in the expression question doesn't always run when you would expect. For instance, if I have 3 questions, A, B and C, located in this order in the survey tree, and question B is an expression question, I would expect question B to run after I advance from question A and before I enter question C. Yet if question B was dynamically joined to the screen of question C, it will run only after I will advance from question C.

To avoid these cases, make sure the following property is enabled in the "Survey Properties" tab:



*** An alternative solution is to define the expression question and the visible question following it to start in a new page (Go to the question's Advanced tab -> enable the "Start a new page" checkbox). This will separate the expression question from the question before it and after it and will make it run when expected.

9 Random Calculations

Always keep the calculations result of random things. For example if you need to randomly generate a number and base a scenario on it make sure to keep that generated number (inside an expression question for example). Another example is if you select X brands to show out of Y keep in a Dummy question the X brands selected. From experience, at some point, someone from your analysis team will query about this data so keep every random calculation outcome you perform during an interview into an exported variable.

10 Remember that text piping is for display only



Scripting Best Practices Guide

“Text piping” is a common method used to dynamically pipe text into a question / answer / topic text during runtime. The piped text is for display only yet it is not saved as part of the survey data. If you’re piping some text that was randomly generated based on some logic and that text is not an existing marked variable in the script - you will need to save it to a variable if you need to know what was the piped in. For example if you have a question with 4 answers that have their text {0} and you pipe into those calculated values or randomly selected items etc. and your question asks for the favorite you will not be able to distinguish in the final data what text was piped to what answer option, unless you’ll save that information in some dummy question (for instance you can have a dummy open ended grid question with 4 topics, matching the 4 answers options, and auto-set each topic with the desired text value in the same order you pipe those texts into the answers options).

11 The Impact of Back

Your surveyors may not normally answer the questions serially as you imagine. They may be going forward and backward during the interview. Always have in mind what will be the impact of going back in your script at every part that you add logic for. For example if you have a Dummy question that is filled in a script of another question do not only set the value but include a code to reset the dummy question (using the ClearAnswer() function), so the setting starts from blank. Not thinking of this and not testing backward scenarios may result in issues during fieldwork.

To learn more about the impact of the back button, view this link:

<http://support.dooblo.net/entries/23850138-The-impact-of-using-the-Back-button-in-interviews>

12 Determine if a question was answered

Under different reasons and scenarios, the logic would need to determine dynamically in the script, if a certain question was answered or not. For example:

Q1 is a single choice question with a brands scale, including a “none” option.

Q2 is a multiple selection question with the same brands scale, and it has this entrance rule:

`!Contains(QRef(1), GetAnswerCount(QRef(1)))`

(Meaning, ask Q2 only if the last answer, which is the “none” answer, was not selected in Q1).

Q3 is a single choice grid question with the brands scale as topics, and it has this code in its start script:

`FilterTopicsByAnswers(CurrQues, QRef(2));`

(Meaning Q3 shows only the brands that were selected in Q2).

So, we would want Q3 to be asked only if Q2 was answered, because if it was not answered, there're no brands to show and Q3 will be displayed "empty" (with no brands on screen).

To determine if a question was answered or not we can use the Answered(x) function (where "x" is the relevant question index). It will return 'true' if "x" was answered (that is if an actual answer was set in this question) or 'false' otherwise.

So basically I can write this in Q3 entrance rule:

`Answered(QRef(2))`

But here's the issue, in the next scenario:

- Q1 was coded with brand index 1.
- Q2 was entered (as expected) and brands indexes 1, 2 and 3 were selected.
- Q3 was entered (as expected).
- Surveyor decided to go back from Q3 by clicking the "Back" button and so went back to Q2.
- Surveyor went back again from Q2 back to Q1 (in this case, Q2 is set as "null response", although the coded brands are still set in it).
- Q1 was changed and instead brand 1, it was coded with the "none" answer.
- Moving forward, Q2 is skipped (due to its entrance rule, as expected).
- Q3 is entered, **NOT as expected**: Q3 is entered because its entrance rule returned 'true', as Q2, although set as "null response", still holds the original brands coded in it, and so it is considered as "Answered". But logically, we would not want Q3 to be asked because Q2 eventually was not asked...

To overcome this, use this function overload:

`Answered(QRef(2), true)`

'true' means to take "null response" questions into account so that this function will return 'false' on "null response" questions, even if they have an actual data set in them.

13 Answer Codes

Each answer has two main attributes: Index and Code (relevant mainly to single choice and multiple selection questions).

The answer index is a serial running number starting with 1, to indicate the place of the answer in the answers list. So the first answer in the list will have index #1, the second answer in the list will have index #2, and so on.

The answer Code is a numerical value that is unique to each answer in the question. This value is the exported value you later see in the exported data (it is the code you see in the selected answer

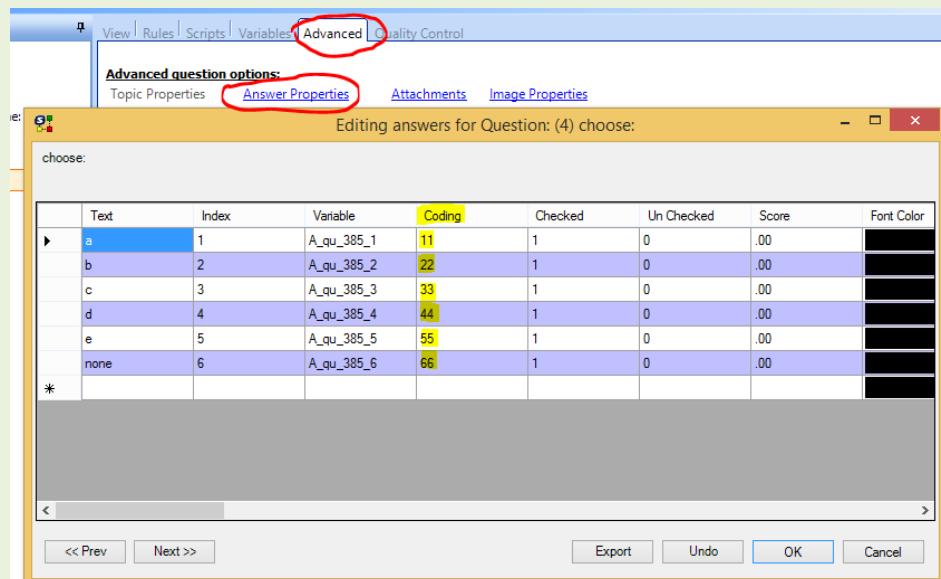
Scripting Best Practices Guide

variable, not the index). By default, the code is the same as the answer index, meaning the code of answer index #1 will be "1", and so on.

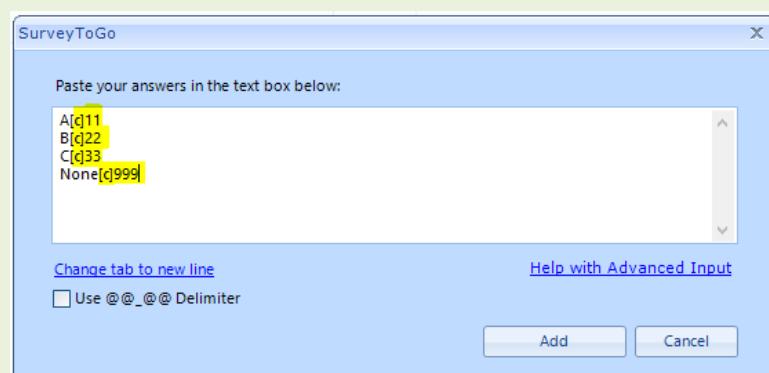
But, there're cases where answer options will have unique codes that are not necessarily the same as the running indexes of the answers. For instance I can have an answer option "None of these" which would be the 5th option in the list but in the questionnaire its code would be indicated as "99". Another example is having a brand named Coca Cola which is third on the brands list but it has a code of 300.

In these cases it is important to set the answer codes to match the codes as in the questionnaire.

You can manually modify the default codes after creating the answers list, through the question's Advanced tab -> Answers Properties:

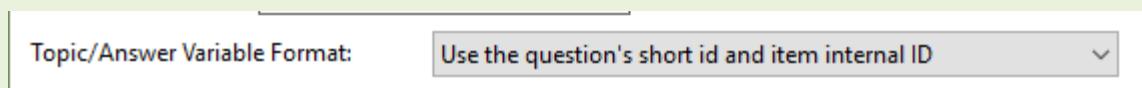


You can also set the answers codes during the answers lists creation: when adding the answers items to the "Multi-Add" window, add "[c]x" at the end of each item, where "x" would be the desired code:



14 Tracker Projects – Keep variable names consistent

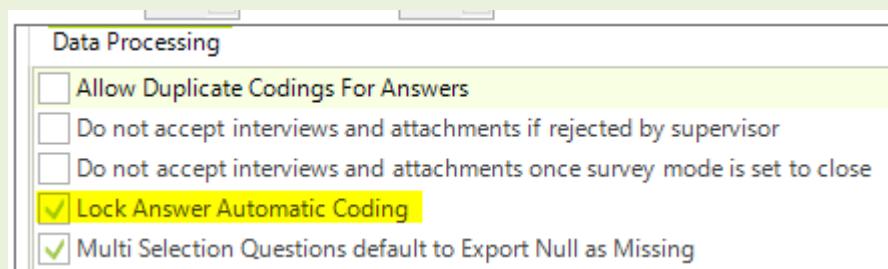
When scripting a tracker project you need to ensure that brands (when used as answers and topics) maintain the same codes and variable names. To ensure it make sure to set the “Topic/Answer Variable Format” to use the short ID and the internal item code (this is defined in the survey’s Advanced tab):



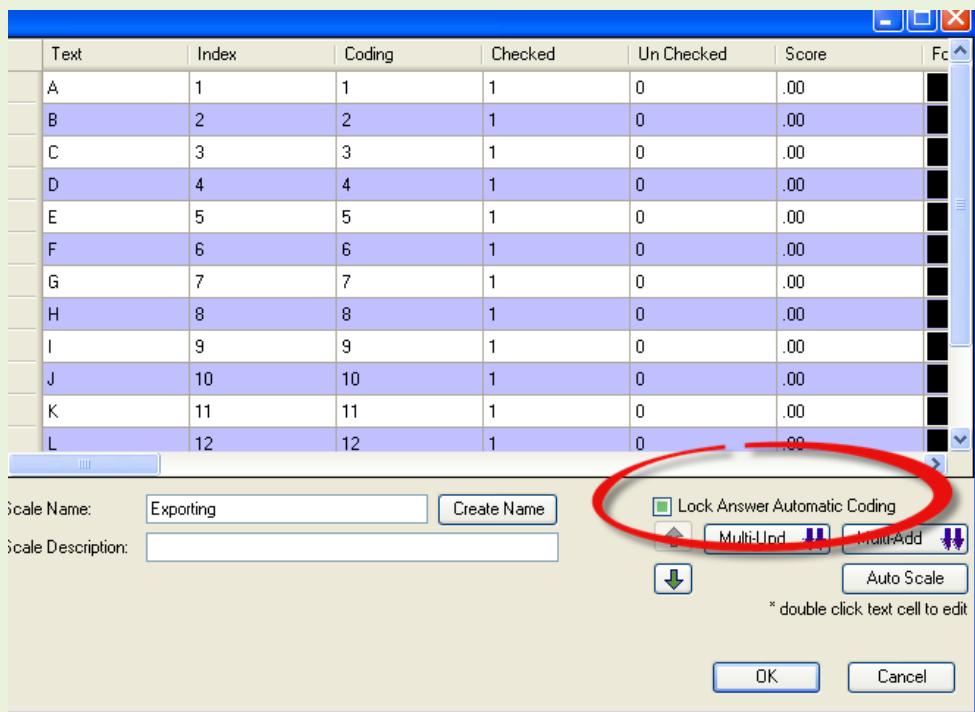
This will ensure that changes in the brand lists or any list in any question for that matter, will maintain the same variable name for a specific brand.

15 Tracker Projects – Keep codes consistent

Like variable names, the items codes should remain consistent between different waves. When updating an answers list between waves (brands etc.) you need to ensure you ‘lock’ the codes before adding new items to the list. This property is enables by default in the Survey Properties tab:



It can also be enabled for each scale individually:



In addition, set the “Use coding format in Iteration export” option in the survey properties so that the code is used to represent iteration on the brands rather than their indexes.

16 Tracker Projects – Update lists without removing / overriding items

Adding new items to a list / scale, should be done by adding them into the end of the list then placing them in the correct position with the Up/Down Arrows and setting their codes in the coding column.

DO NOT remove items and add new ones instead, as it will generate new variable names and internal IDs, thus it won't be consistent with previous waves.

If using the “Multi-Update” button to update a list in bulk – make sure not to override any item incorrectly. The “Multi-Update” feature just updates the texts of the existing items (based on the same indexes order). So for example, if I have this list:

A (index 1)
B (index 2)
C (index 3)

And I use the “Multi-Update” with this list:



Scripting Best Practices Guide

New A

B

New C

Once done, the list will still have 3 options, with the same indexes, codes and variable names, but with these labels:

New A

B

New C

So in trackers, where the data should stay consistent, but there are changes to a list that need to be done and you want to use the “Multi-Update” to do so, make sure to do it correctly to avoid data inconsistency. For example, if we again have this list:

A

B

C

And in the current tracker script we are now requested to change the list to be:

A

New C

(That is, basically exclude B from the list and keep the option C just change its text)

If I'll use the “Multi-Update” and wrote this list:

A

New C

That would be wrong as the “New C” label will override the option in the second index, “B”, meaning the code and variable name that were originally linked to “B”, will now be shown in the data for “New C” and that's not what we want. The right way (other than perform this all manually without the “Multi-update”...) would be to write the full list in the “Multi-update” feature like this:

A

B



Scripting Best Practices Guide

New C

And later just mark option “B” as hidden.

17 “Null response” questions

A “null response” question is a question that was not answered, that is a question with no answer set in it. In the exported data “null response” questions will be exported with the missing value (“-1” by default unless changed).

By default, before a question is answered in an interview, it is considered as “null response”.

Also, if a certain question is skipped in the interview and never answered (due to jump rules, entrance rules or any other reason) it will stay as “null response”.

Another case when a question is set as “null response” is when a question is displayed on screen and the “Back” button in the app is clicked. Meaning, when you go back from a question – that question will be set as “null response”. If that question was originally answered (before going back), the data will still be saved in the question “back-stage” but it won’t be shown in the final data as long as the question is set as “null response”.

18 “Null response” answers/topics

By default, once an answer/topic is set as not visible before the question is displayed, it is considered as a missing value. That means that even if this answer / topic automatically set through script before entering the question, and in addition it was also set as hidden before entering the question – it will be considered as “null response” and won’t show the value that was automatically set in it (it will be shown with the missing value instead).

To overcome this – if you have an answer / topic that you want to auto-set through the script but not display it on screen: perform the auto-setting AFTER the question is displayed, for instance do it from the question’s End script.

19 Review the data before going live

It is extremely important to have your DP review the data before the script goes into production. You may find out only after the entire field work ends that the DP has issues with the data structure. Make sure to perform several test interviews and export their data to hand over to your

DP for review. Another method is to generate Dummy Data using the Dummy Data generator:
<http://support.dooblo.net/entries/23000471-Using-the-SurveyToGo-Dummy-Data-Generator>

20 Exported questions

By default, each question created in the script is "exportable" (except for Empty questions types), meaning it will be included in the exported data.

In some cases, there would be certain questions that can be excluded from the export if they're not needed for the actual data analysis (for instance all kinds of "help questions" such as dummies, certain expressions etc.). In these cases, it is better to exclude those questions from the export in advance, to avoid an unnecessary large data structure.

To exclude a question from the export, go to the question's Variables tab and disable the "Export" checkbox:

